UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informatica

TRABAJO FIN DE GRADO

Mini-Robots móviles radiocontrolados para aplicaciones colaborativas

Autor: Eva Garcia Ocaña
Tutor: Guillermo José González de Rivera Peces

junio 2020

Eva Garcia Ocaña
Mini-Robots móviles radiocontrolados para aplicaciones colaborativas

Eva Garcia Ocaña

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

Quisiera agradecer en primer lugar a mi tutor, Guillermo González de Rivera, el haberme guiado a lo largo de este proyecto. También me gustaría agradecer a la Universidad y en especial a la EPS el haber podido realizar este proyecto.

Agradezco a Daniel López apoyarme y animarme siempre Agradezco a David Garcia toda la ayuda que me ha prestado.

Agradezco a mi familia por estar siempre ahí cuando los necesito.

Resumen

En la actualidad, existen numerosas tareas que el ser humano no puede realizar o realiza con suma dificultad debido a factores como puede ser la accesibilidad o peligrosidad. Muchos de estos problemas podrían resolverse mediante la robótica coordinada.

El objetivo del proyecto es la creación de una aplicación de robótica coordinada que consistirá en un conjunto de robots de tamaño reducido controlados mediante un coordinador y conectados entre sí mediante una red. Estos robots permitirán permiten continuar la línea de robótica colaborativa, adaptándola a situaciones donde fuera necesario tener un robot de dimensiones reducidas.

En este proyecto, se ha diseñado un grupo de robots radiocontrolados. Estos robots son capaces de actuar de forma coordinada siguiendo instrucciones de manera remota. Para el desarrollo de los robots se empleo el software de diseño de PCBs Altium designer. Se crearon 2 versiones, una primera de mayores dimensiones para comprobar que todo funcionase y otra de menores dimensiones.

En este documento se detallan los distintos pasos seguidos tanto en el diseño como para el desarrollo del proyecto, de forma que también pueda servir como documento de referencia para la creación de proyectos similares.

Palabras clave

Robótica, Robots radiocontrolados

Abstract

Nowadays, there are several tasks that human being cannot carry out or carry out with difficulty due to factors such as accessibility or dangerousness. A lot of this issues would be solved through coordinated robotics.

The objective of this project is the setting up of a coordinated robotics application consisting of a group of small size robot controlled by a coordinator and connected via network. These robots would allow future researching in collaborative robotics field, adapting it to situations in which it is necessary to have a small size robot.

For this project, a group of radio-controlled robots has been designed. These robots can act in a coordinate way following remote instructions. For the design of the robots, Altium Designer was used. Two versions were created, a first with bigger dimensions to ensure that everything works and another of smaller dimensions.

In this document, the different steps of the design are detailed, so this document can also be a referent for the creation of similar projects.

Keywords

Robotics, Radio controlled robots

Índice

1 Introducción **1**

1.1 Motivación 1

1.2 Objetivos 1

1.3 Organización de la memoria 2

2 Estado del arte **5**

2.1 Aplicaciones colaborativas 5

2.2 Tecnologías empleadas 6

3 Diseño **9**

3.1 Requisitos 9

3.2 Diseño 11

4 Desarrollo **15**

4.1 Desarrollo de la red 15

4.2 Desarrollo del coordinador 17

4.3 Desarrollo del robot 18

5 Integración, pruebas y resultados **29**

5.1 Pruebas del robot 29

5.2 Pruebas del coordinador 30

5.3 Pruebas conjuntas 31

6 Conclusiones y trabajo futuro **33**

6.1 Conclusiones 33

6.2 Trabajo futuro 33

Bibliografía **35**

Apéndices **37**

A Listado de componentes de la versión 1.0 **39**

B Listado de componentes de la versión 1.1 **41**

C Guía de montaje **43**

D Guía de instalación del software del robot **47**

E Guía de instalación de las librerías **49**

F Esquemático de la version 1.0 **51**

G PCB de la versión 1.0 **53**

H Esquemático de la Versión 1.1 **55**

I PCB de la versión 1.1 **57**

J Código de los robots **59**

K Código del coordinador **65**

Listas

Lista de figuras

2.1 Estructura Zigbee 7
2.2 Analog Discovery 2 y WaveForms. 8

3.1 Diagrama Red 11
3.2 Diseño Coordiandor 12

4.1 Configuracion Inicial 16
4.2 Parametos Coordinador 16
4.3 Parametos router 16
4.4 Parametos Red 17
4.5 Informacion AI 17
4.6 Diagrama de clases 19
4.7 Esquemático Xbee 21
4.8 Esquemático Motores 22
4.9 Esquemático USB 22
4.10 Esquemático sensor óptico 22
4.11 Esquemático portapilas 23
4.12 Esquemático conversor de voltaje 23
4.13 Footprint Xbee 24
4.14 Placa impresa 25
4.15 Logo 25
4.16 Esquematico 1.1 26
4.17 Comparativa Tamaños 27
4.18 Modelo versión 1.1 27

5.1 Robot 30

C.1 Controlador de motor 43
C.2 Motor junto a encoder 44
C.3 Diseño de la placa superior. 44
C.4 Diseño de la placa inferior. 44

D.1 Menu terminal 47

F.1 Esquematico 1.0 52

G.1 Diseño PCB Versión 1.0. 54

H.1 Esquematico inferior 55
H.2 Esquematico 56

I.1 Diseño del PCB superior. 57
I.2 Diseño del PCB inferior. 58

Lista de tablas

2.1 Especificaciones del xbee. 8

3.1 Resumen de los requisitos 10
3.2 Correspondencia de los requisitos con los componentes 13

A.1 Listado Componentes versión 1.0 40

B.1 Listado Componentes Versión 1.1 41

C.1 Correspondencia identificadores placa inferior 45
C.2 Correspondencia identificadores placa superior 45

1

Introducción

1.1. Motivación

La realización de este trabajo viene motivada por el deseo de crear una aplicación de robótica colaborativa, si bien en este proyecto solo se crearan las bases. Esta clase de sistemas pueden significar un gran avance en cualquier industria, desde la limpieza o descontaminación de lugares, inaccesibles para el ser humano tanto por tamaño como por peligros varios, hasta cadenas de montaje. Estos equipos coordinados pueden ser de gran ayuda para infinidad de tareas, algunas incluso por descubrir. De esta forma, el tener un equipo de robots radiocontrolados, ampliara las posibilidades de investigación robótica del grupo de investigación donde se está realizando este proyecto. También se incluye la motivación propia del alumno de aprender las bases de la robótica, siendo este un campo con grandes aplicaciones.

La creación de uno de estos equipos desde cero puede resultar complicado, pero permite el aprendizaje de todos los procesos necesarios para su fabricación.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Grado es el diseño y ensamblaje de mini robots controlados por radio con la capacidad de colaborar entre ellos. La idea fundamental es que sean capaces de comunicarse y organizarse para resolver de esta forma problemas que por sí solos no podrían. Este proyecto se encargará de la creación de estos robots, dejando la realización de aplicaciones complejas a proyectos futuros, por lo que también se incluye como objetivo adicional que todos los componentes de este trabajo permitan su modificación o expansión futura de forma simple. De esta forma, el objetivo de este proyecto es crear una base sobre la que se podrá desarrollar aplicaciones colaborativas, pero no la creación de una aplicación colaborativa como tal.

Por ende, se tendrán que crear dispositivos que deberán de ser capaces de moverse y actuar siendo controlados de manera remota, pero ninguna aplicación específica para ellos. También se incluye como

objetivo el aprendizaje por parte del alumno de los procedimientos para diseñar y construir un robot, así como el aprendizaje del uso de las herramientas para ello.

1.3. Organización de la memoria

A continuación, se detallará la organización de este documento, listando todos los apartados y detallando su contenido.

Introduccion. En este primer apartado, se informan de la motivación y los objetivos de este proyecto. Así mismo se define la estructura y organización de esta memoria.

Estado del arte. Esta sección se divide en dos partes, por un lado, se investigan proyectos de robótica colaborativa o de enjambre y por otro lado se describen las tecnologías que se emplearan para desarrollar este proyecto.

Diseño. En esta sección se divide en dos. En primer lugar, se definen los requisitos del proyecto a partir de los objetivos y, en segundo lugar, se realiza un diseño en base a estos requisitos. Tanto los requisitos como el diseño se dividen en tres, la red, el coordinador de esta y los robots.

Desarrollo. En esta sección se desarrolla la implementación de los requisitos y el diseño. Es de vital importancia, pues es donde se explican las decisiones tomadas. Se divide en tres subsecciones, una para el desarrollo de la red, otra para el desarrollo del coordinador y otra para el desarrollo de los robots.

Integración, pruebas y resultados. En esta sección se describen las pruebas realizadas para asegurar el correcto funcionamiento del sistema y los resultados de estas.

Conclusiones y trabajo futuro. Esta sección recoge las conclusiones extraídas durante la realización del proyecto y se reflexiona sobre el posible trabajo futuro.

Referencias. Listado de todas las fuentes consultadas durante la realización de este proyecto.

Apéndices

Apéndice A. Listado de componentes de la versión 1.0.

Apéndice B. Listado de componentes de la versión 1.1.

Apéndice C. Guía de montaje.

Apéndice D. Guía de instalación del software del robot.

Apéndice E. Guía de instalación de las librerías.

Apéndice F. Esquemático de la versión 1.0.

Apéndice G. PBC de la versión 1.0.

Apéndice F. Esquemático de la versión 1.1.

Apéndice G. PBC de la versión 1.1.

Apéndice G. Código de los robots.

Apéndice H. Código del coordinador.

2

Estado del arte

En esta sección se muestran algunos ejemplos de sistema similares y se detallan algunas de las tecnologías empleadas en este trabajo.

2.1. Aplicaciones colaborativas

El uso principal de este tipo de robots es el de aplicaciones colaborativas, un conjunto de robots que se coordinan y cooperan para logar un objetivo común. El grado de interacción y de autonomía de los robots puede variar, permitiendo redes centralizadas en un único coordinador que dirija a los demás o redes donde cada individuo actúa de forma independiente, interactuando con el resto para logar la meta.

En esta sección se mostrarán varios ejemplos de robótica colaborativa.

2.1.1. Swarm-bot y Swarmanoid

Swarm bot es un proyecto financiado por el programa “futuro y tecnologías emergentes” de la comisión europea. Consistió en la creación de bots auto ensamblables y autoorganizados. Además de con capacidad de movimientos, también contaban con un amplio conjunto de distintos sensores (de proximidad, lumínicos, humedad, etc.) y con mecanismos para comunicarse (luces led y micrófonos) e interactuar (brazos con pinzas) con sus similares. El proyecto también incluía una torre con luces que podía desempeñar varios roles dentro de los experimentos. [1]

Para comprobar las capacidades de los robots, se diseñó una prueba que consistía en el desplazamiento de un objeto por un terreno con obstáculos, de esta forma había dos desafíos; en primer lugar, encontrar un camino hasta el objetivo y en segundo lugar recorrer ese camino empujando el objeto de forma coordinada. Los robots fueron capaces de realizar ambas tareas. Para poder realizar los experimentos, se crearon 2 áreas de prueba para los robots (una en Lausana y otra en Bruselas) que se modificaban según requiriera la prueba a desarrollar.

Posteriormente y a raíz de este proyecto, surgió el proyecto Swarmanoid, que llevaba un paso más el concepto, creando diversos tipos de robots para la realización de diversas tareas. [2]

2.1.2. Robotarium

El robotarium es un proyecto puesto en marcha por el Instituto Tecnológico de Georgia. El proyecto consiste en un enjambre de robots de acceso remoto a la que cualquiera puede acceder. Generalmente, este tipo de sistemas tienen altos costes de operación y mantenimiento, así que el objetivo del robotarium es facilitar el acceso, tanto a profesores como a alumnos, a este tipo de sistemas. Debido al libre acceso, consta de un sistema de seguridad para evitar colisiones o malos usos que puedan dañar los robots. [3] También dispone de un simulador, disponible en Matlab o en python, que permite verificar el código escrito. Previamente a realizar cualquier experimento, este será simulado.

2.1.3. Kilobot

La universidad de Harvard creó un sistema de bajo coste, logrando que el precio por unidad fuera inferior a 15 dólares y con un tiempo de construcción de 5 minutos. Así mismo, este sistema se ideo para que una sola persona pudiera realizar operaciones con una gran cantidad de robots. Los kilobots pueden moverse, comunicarse entre ellos y recibir instrucciones de forma externa. También disponen de luces led que pueden encender y apagar. Miden menos de 5cm, tanto de alto como de ancho. [4]

2.2. Tecnologías empleadas

A continuación, se describen las tecnologías y herramientas empleadas en el desarrollo de este proyecto de fin de grado. Su uso y aprendizaje también forma parte del proyecto.

2.2.1. Zigbee

Zigbee es un conjunto de protocolos de comunicación para redes de corto alcance. Está basado en el estándar IEEE 802.15.4. Fue diseñado para permitir un consumo bajo que maximice la vida de las baterías. Opera en un ancho de banda de 2.4 GHz con 16 canales. Permite cifrar los paquetes con aes 128 y tiene un alcance aproximado de 100m y permite la trasmitir a una velocidad de 20 a 250 kbps. [5]. La figura 2.1 muestra las distintas capas del protocolo zigbee y su relación con el estándar IEEE 802.15.4.

A demás, es un estándar ampliamente utilizado por la industria lo que permite que todo tipo de dispositivos lo incorporen. Hay más de 300 millones de dispositivos certificados que lo usan, incluyendo

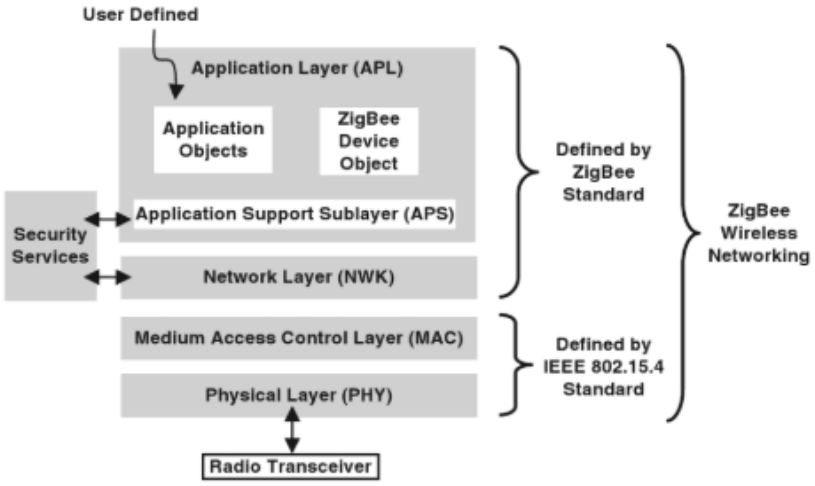


Figura 2.1: Imagen sacada de ZigBee Wireless Networks and Transceivers

sensores, interruptores, controles remotos, sistemas de iluminación, sistemas de domótica, entre otros. [6]

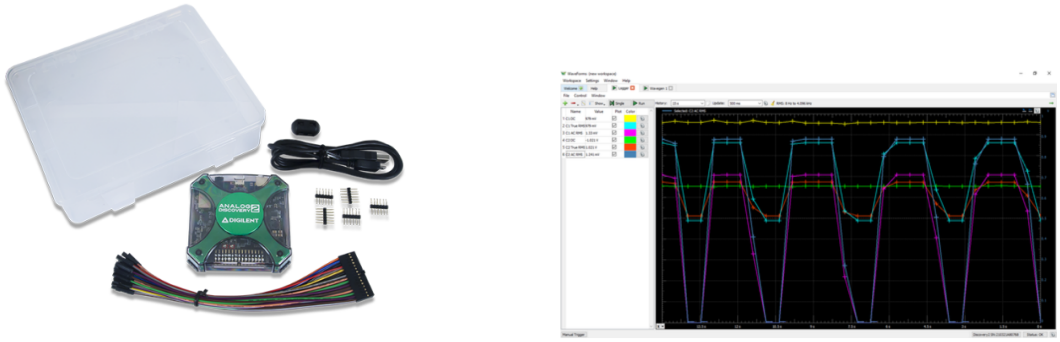
Dentro del protocolo se distinguen tres roles para sus participantes, en primer lugar, estaría el coordinador, que crea la red y permite al resto de dispositivos unirse a ella. Luego estarían los dispositivos router, que permiten expandir la red y que se conectasen a ellos otros dispositivos. Por último, estarían los dispositivos finales, que tienen la capacidad de hibernar por periodos de tiempo para así consumir lo menos posible. El dispositivo al que estuvieran conectados almacenara los mensajes que tuvieran que recibir durante su hibernación para transmitírselos cuando despierten. A diferencia de los otros dos, los dispositivos finales no admiten que se conecte ningún dispositivo a ellos.

2.2.2. Altium Designer

Altium Designer es un software de diseño de esquemáticos y circuitos impresos. Esta desarrollado por la empresa Altium Limited. Se seleccionó este programa debido a su versatilidad. Permite el diseño de componentes y de documentación autogenerada. También permite el diseño y simulación de FPGA.

2.2.3. Analog Discovery 2 y WaveForms

Analog Discovery 2 es un osciloscopio digital que se conecta por usb al ordenador. Permite la generación de funciones y también posee entradas analógicas y digitales. Se complementa con el software WaveForms, que permite la visualización de datos y la generación de ondas. Algunas de sus funciones más destacadas son: osciloscopio, generador de funciones, voltímetro y fuente de alimentación, entre otras.



(a) Interfaz de WaveForms

(b) Dispositivo Analog Discovery

Figura 2.2: Analog Discovery 2 y WaveForms, ssacadas de la web del fabricante [7].

2.2.4. Digi Xbee 3 y XCTU

Los dispositivos digi xbee 3 son módulos de radio programables, están desarrollados por la empresa Digi, especializada en dispositivos remotos e IoT. Trasmiten a una frecuencia de 2.4Gh y admiten varios protocolos de radio (zigbee, digi mesh e IEEE802.15.4). También soportan conexión por bluetooth. La tabla 2.1 muestra sus características principales [8].

XCTU es un software proporcionado por el mismo fabricante de los dispositivos. Permite la actualización, configuración y programación de los dispositivos.

Rango	Hasta 60m en interiores, hasta 1200 en exteriores.
Frecuencia	ISM 2.4 GHz.
Velocidad de transferencia	250 Kbps
Protocolo	Zigbee.
Canales	16.
Voltaje de alimentación	2.1 a 3.6V.

Tabla 2.1: Especificaciones del xbee.

3

Diseño

En esta sección se expondrán, en primer lugar, los requisitos iniciales del proyecto y posteriormente se detallarán las decisiones de diseño que se han tomado.

3.1. Requisitos

El objetivo del proyecto es la creación de las bases de un sistema de robótica colaborativa, es decir la creación de robots controlados remotamente. De esta forma, habrá que crear robots, un coordinador que los dirija y una red que les permita comunicarse. Tanto el diseño y requisitos como el desarrollo se dividen para atender a estas tres categorías. A continuación, se detallan estos requisitos, la tabla 3.1 muestra un resumen de los requisitos.

3.1.1. Requisitos de red

Se necesitará una red inalámbrica que permita una fácil expansión, puesto que se trabajará con un número indeterminado de unidades móviles. Así mismo, debe de permitir un bajo consumo para alargar la vida de las baterías de estas unidades. El coordinador debe de poder comunicarse de forma bidireccional con los dispositivos, pero no se incluye la posibilidad de que los dispositivos se comuniquen entre ellos. Debe de permitir que los robots actúen de repetidores.

3.1.2. Requisitos del coordinador.

El coordinador tiene el objetivo de organizar a los robots. De esta forma debe de poder enviar instrucciones a los mismos, tanto individualmente como colectivamente, también debe de poder recibir información de los robots si esta es solicitada. Sera controlado por el usuario, pero también se incluye la posibilidad de operar de forma preprogramada o autónoma. También debe de poder crear y gestionar la red.

3.1.3. Requisitos del robot

Los robots serán unidades móviles. Por ello deben de contar con algún sistema de tracción, así mismo será necesario que disponga de baterías portátiles para tener autonomía. Deben de poder conectarse a la red y enviar y recibir mensajes. Deben de poder ejecutar las instrucciones que reciban del coordinador y responderle si les solicita información. Deben de poder recabar información del medio. También deben de poder actuar como repetidores, para permitir expandir la red y no depender únicamente del alcance del coordinador. Para facilitar su uso se contempla que puedan conectarse a un ordenador con el fin de modificar su programación o parámetros.

Red
Debe de permitir la comunicación bidireccional de los robots con el coordinador.
Debe de tener un bajo consumo.
Debe de ser escalable.
Coordinador
Ser capad de crear y gestionar la red.
Tiene que poder enviar y recibir mensajes a través de la red.
Tiene que permitir mandar instrucciones a los robots, tanto de forma individual como colectiva.
Debe de poder actuar tanto de forma autónoma (preprogramada) como controlado.
Robot
Tienen que poder moverse.
Deben tener autonomía (batería).
Tienen que recibir algún tipo de información del medio.
Tienen que poder conectarse a la red, enviando y recibiendo mensajes.
Debe de poder ejecutar las instrucciones que el coordinado mande.
Debe de poder informar al coordinador de la información que obtenga, cuando le sea solicitada.
Deben de poder actuar como repetidores, expandiendo la red.
Permitir la configuración a través de un ordenador.

Tabla 3.1: Resumen de los requisitos

3.2. Diseño

Una vez definidos los requisitos básicos, se definirán los distintos aspectos tenidos en cuenta en el diseño. En la sección 4 se desarrolla el diseño.

3.2.1. Diseño de la red

Varios aspectos fueron tenidos en cuenta a la hora de elegir qué forma tendría la red y que protocolo debía usarse. En primer lugar, se decidió la forma de la red. La red tendrá topología de malla, puesto que, aunque la mayor parte de comunicaciones se realicen entre un robot y el coordinador, también es posible que los robots actúen de repetidores. Véase la figura 3.1.

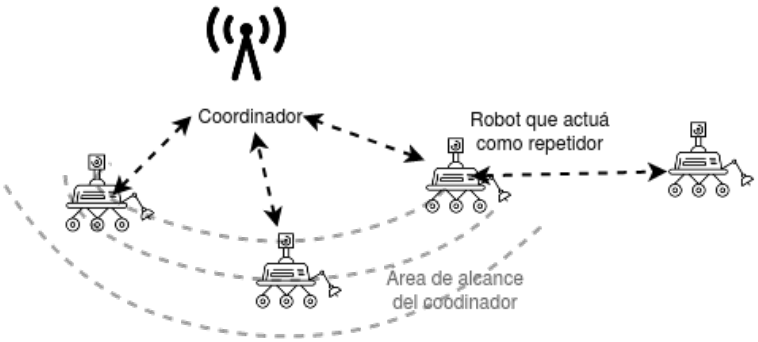


Figura 3.1: Se muestra como un robot expande la red, actuando de repetidor

La red tenía que cumplir tres requisitos básicos: ser expandible, permitir la comunicación y que dispositivos actuasen de repetidores y permitir un bajo consumo, por ello se eligió el protocolo Zigbee, puesto que cumple todos estos requisitos y se estructura en forma de malla. Además, al ser ampliamente utilizado da acceso a un gran número de dispositivos que lo incorporan, lo que da una gran libertad a la hora de elegir cuál usar en este proyecto.

3.2.2. Diseño del coordinador

El diseño del coordinador está dividido en dos capas. La primera capa se encargar del control más directo de la red y los dispositivos, mientras que la segunda permitirá al usuario interactuar con la red.

Por un lado, la primera capa gestiona el propio dispositivo de radio, generando una interfaz que permite el escaneo de la red, así como el envío y recepción de mensajes. Se encarga de inicializar el dispositivo al iniciar y liberarlo al terminar.

La siguiente capa es una línea de comandos, que transforma los comandos dados en llamadas a la capa inferior. Esta capa permite al usuario enviar instrucciones a los distintos dispositivos. La aplicación

permitirá tanto la entrada de comandos por teclado como la lectura de un fichero.

Tal y como se muestra en la figura 3.2, el usuario escribiría un comando, el intérprete lo transformará en una llamada al módulo de radio y este lo enviará a través de la red. Si fuera necesario el robot o robots que lo recibieran contestarían y la respuesta se mostraría al usuario.

Esta arquitectura por capas permite la reutilización de código y facilita el desarrollo de aplicaciones futuras, permitiendo modificar la capa superior manteniendo la inferior.

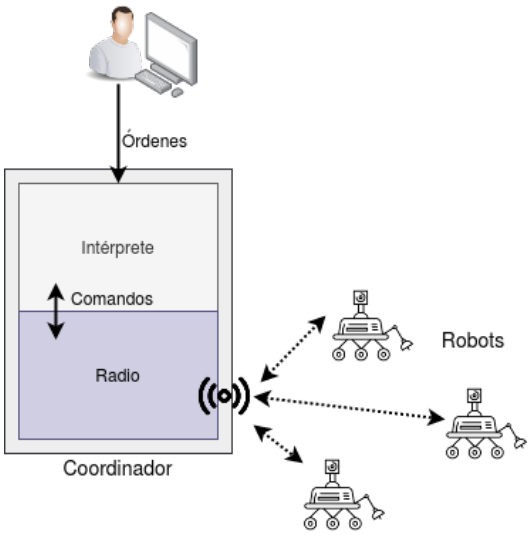


Figura 3.2: Muestra el diseño del coordinador

3.2.3. Diseño del robot

A la hora de diseñar un robot dos aspectos tienen que ser tenidos en cuenta, por un lado, el hardware sobre el que va a operar, que va a definir las capacidades básicas del mismo, y por otro el software, que va a refinar esas capacidades. A continuación, se explicará de forma detallada el diseño de cada una de estas partes.

Diseño del software

Como se definió en los requisitos, el robot debe de ser capaz de recibir instrucciones y operar en función a ellas. La siguiente lista muestra las instrucciones básicas a las que debe de ser capaz de reaccionar:

- Moverse hacia delante
- Moverse hacia atrás
- Girar a la derecha
- Girar a la izquierda
- Cambiar la velocidad o pararse

- Indicar el estado de los sensores

Estas instrucciones corresponden al cumplimento de los requisitos

Diseño del hardware

Antes de diseñar el esquemático o el circuito, se decidió que componentes habrían de llevar los robots. Al ser robots pequeños, el tamaño es limitado y había que seleccionar bien los componentes. Algunos como los motores, el dispositivo de radio o la batería eran evidentes. Además, se decidió añadir un puerto USB, sensores ópticos y encoders para tener información del movimiento de las ruedas. Adicionalmente, se pensó en alguna forma de hacer el diseño expandible, por lo que se decidió que se facilitaría de alguna forma la instalación de componentes extra, para permitir la creación de aplicaciones futuras.

El diseño del dispositivo se desarrolló en Altium, el aprendizaje de esta herramienta de diseño de pcbs formó parte de este trabajo de fin de grado.

La tabla 3.2 muestra un listado con los componentes elegidos y los relaciona con los requisitos de la tabla 3.1

Componentes	Requisitos
Motores	Tienen que poder moverse.
Portapilas	Deben tener autonomía.
Sensores Ópticos, encoders	Tienen que recibir algún tipo de información del medio.
Microprocesador	Ejecutar instrucciones e informar de la información que obtenga.
Módulo de radio	Conectarse a la red, enviando y recibiendo mensajes.
Conexión usb	Permitir la configuración a través de un ordenador.

Tabla 3.2: Correspondencia de los requisitos con los componentes

4

Desarrollo

En este capítulo se detallan los distintos aspectos del desarrollo del proyecto, incluyendo la creación de la red, del coordinador, de los dispositivos y del software. Los apéndices D y E complementan la información aquí dada, informando sobre como instalar el software y las librerías necesarias.

4.1. Desarrollo de la red

En esta sección se hablará de como configurar los dispositivos XBee para el correcto funcionamiento de la red. Estos dispositivos fueron elegidos puesto que permiten unificar el módulo de radio y el microprocesador en uno, permitiendo usar un único componente y ahorrar recursos. Estos dispositivos actúan como microprocesadores y radio en los robots y como emisor de radio en el coordinador. Toda la configuración se realizó a través de XCTU.

4.1.1. Configuración de los microprocesadores

La configuración de los dispositivos se realizó en XCTU, un software desarrollado por el fabricante de los mismos para su configuración. A continuación, se detallan los pasos a seguir para configurar los dispositivos. El protocolo Zigbee admite tres roles en la red, coordinador, router y end device. Para este proyecto solo se usarán dos, coordinador y router, pues nos interesa que los robots puedan redirigir los mensajes. De esta forma, en esta guía solo se explicará como configurar esos dos tipos. El programa ofrece la posibilidad de establecer multitud de parámetros, pero nos centraremos únicamente en los imprescindibles para crear una red. El programa dispone de una barra de búsqueda en la parte superior que permite ir rápidamente a cualquier parámetro. En la figura 4.1 se muestra la configuración por defecto de un dispositivo. Antes de iniciar la configuración, se recomienda actualizar el dispositivo y cerciorarse de que esta en modo zigbee. Estas dos tareas se pueden realizar a través de la función *update* del mismo programa.

Para configurarlo como coordinador es necesario modificar el parámetro CE y asignarle el valor 1 (“Form Network”). También, aunque no obligatoriamente, es recomendable asignar un valor de ID, este

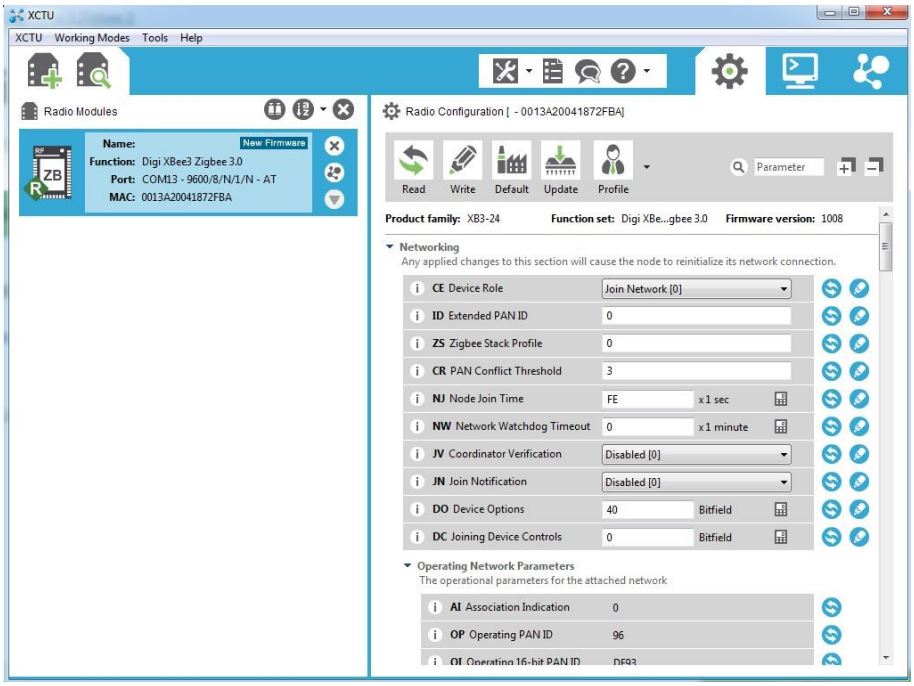


Figura 4.1: Configuración del dispositivo.

valor será el id de la red. Si se deja en 0 el propio dispositivo elegirá uno aleatoriamente. Todo el resto de parámetros se pueden dejar con su valor por defecto. En la figura 4.2 se muestra la configuración de los parámetros.

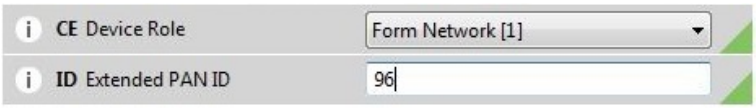


Figura 4.2: Parámetros CE e ID.

Para configurar un dispositivo como router, es necesario poner el parámetro CE a 0 (“Join Network”). Al igual que con el coordinador, es recomendable asignar un valor de ID, pero si se deja a 0 se conectará a cualquier red que encuentre. también es recomendable habilitar la verificación de coordinador (JV) cambiando su valor a 1. De esta forma al conectarse a una red se cerciorará de que esa red tiene coordinador. En la figura 4.3 se muestra la configuración de los parámetros.

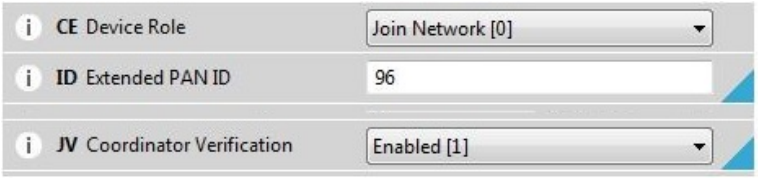


Figura 4.3: Parámetros CE, ID, JV

Adicionalmente, se les puede poner un nombre modificando el parámetro NI. De esta forma, es más

fácil identificar los distintos dispositivos. Una vez que se ha configurado el coordinador y, al menos, un router, es posible ver el estado de la conexión desde el propio programa. El parámetro AI indica el estado de la misma, y el parámetro OP indica el id de la red a la que se ha conectado. Si todo está bien, El valor de AI será 0 y el valor de OP será el id de la red.

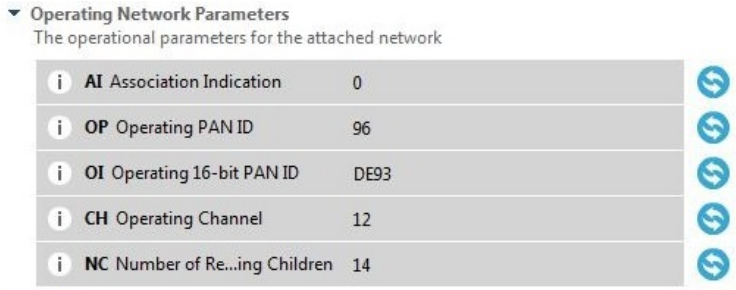


Figura 4.4: Parámetros de la red

Si hubiera algún fallo, el parámetro AI indicaría que fallo es. Es posible ver que representa cada valor accediendo a la información del parámetro. Si se hace click en la i de cada parámetro, se despliega información sobre el mismo (que valores puede tomar, que significa, etc.). La figura 4.5 muestra la información correspondiente al paramento AI.

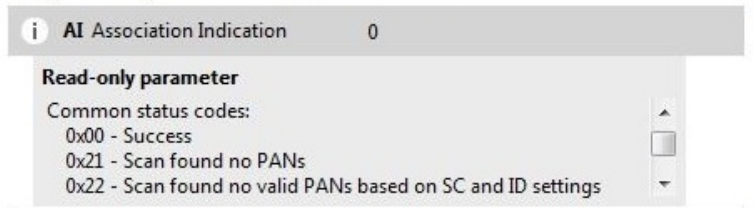


Figura 4.5: Informacion del parametro AI

4.2. Desarrollo del coordinador

Como coordinador se empleó un dispositivo Digi xbee3 conectado a un ordenador. El software se realizó en Python, empleando librerías del fabricante, el apéndice K contiene el código completo y el apéndice E la guía de instalación de las librerías. Para la primera capa, se creó la clase “NetworkCoordinator”, que se encarga de realizar las tareas de red. Dentro de esta clase los siguientes métodos fueron implementados:

- Initialize
- __data _received _callback
- scan _network
- discover _network
- get _devices
- send

- broadcast
- close
- atm _command

Sobre esto se creó otra capa, una línea de comandos. Esta interfaz permite al usuario escribir comandos, que se enviarán y serán ejecutados por los robots.

Los comandos implementados para la línea de comandos son los siguientes:

- Send – envía un comando a un robot, uso: “send [nombre del robot] ‘[comando] [parámetro del comando]’”
- Broadcast – envía un comando a todos los robots, uso: “send ‘[comando] [parámetro del comando]’”
- List – muestra una lista de con todos los dispositivos.
- Scan – escanea la red para descubrir todos los dispositivos conectados a ella.
- Help – muestra todos los comandos posibles.
- Wait – espera el tiempo en segundos indicado, uso: “wait [tiempo]”

El uso del coordinador es el siguiente: python coord.py [nombre fichero]. Si se indica un archivo ejecutará las instrucciones del mismo y si no ejecutará los comandos que se le escriban.

4.3. Desarrollo del robot

El diseño general del dispositivo se basa en un módulo Xbee, que sirve como microprocesador y módulo de radio. Además, se incluyeron motores, enconders y sensores ópticos, para permitir la movilidad y extraer información del entorno. Antes de diseñar el esquemático o el circuito, se decidido que componentes habría de llevar cada robot. Al ser robots pequeños, el tamaño es limitado y había que seleccionar bien los componentes. Algunos como los motores, el dispositivo de radio o la batería eran evidentes. Además, se decidido añadir un puerto USB, sensores ópticos y enconders para tener información del movimiento de las ruedas. El diseño del dispositivo se desarrolló en Altium.

4.3.1. Desarrollo Software

Como parte del trabajo llevado a cabo dentro del presente proyecto, se desarrolló el software de control de los robots, este software se hizo en micropython, una versión de Python diseñada para dispositivos con baja potencia. El apéndice J muestra el código completo. Para la versión 1.0, se crearon las clases Motor, OpticalSensor y Encoder para representar estos componentes y gestionar su funcionamiento, así mismo se creó la clase Robot para contenerlas. Para la versión 1.1 se añadió la clase Led, para controlar las luces led. La figura 4.6 muestra el diagrama de clases.

El funcionamiento básico del robot se basa en un bucle infinito, dentro del cual se comprueba si hay nuevos mensajes. Si los hubiera se encargaría de ejecutarlos. En último lugar, independientemente de

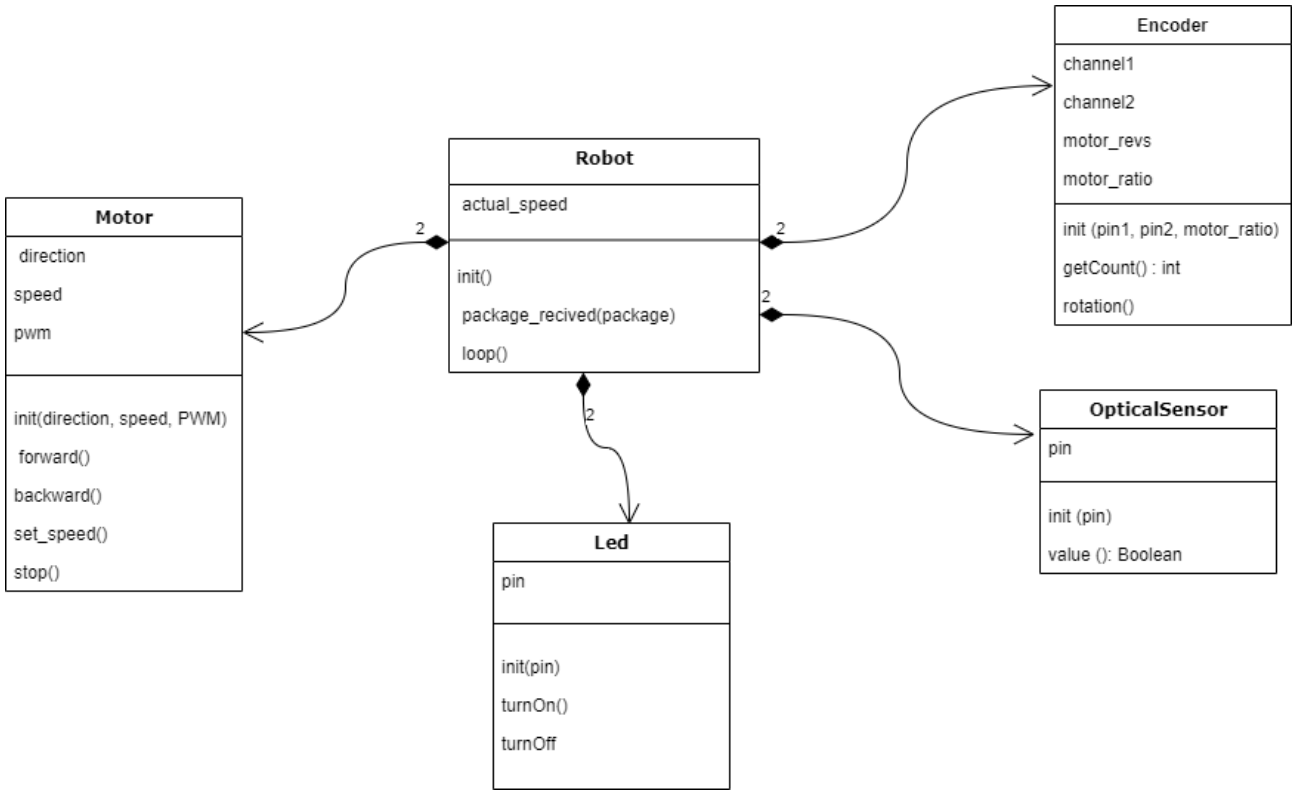


Figura 4.6: Diagrama de clases, la clase led solo está disponible para la versión 1.1 del dispositivo.

si ha habido mensajes o no, comprueba los encoders, para llevar la cuenta del giro de las ruedas y así poder medir distancias.

Los comandos que admite son:

- “forward [velocidad]” – El robot se desplaza hacia delante, se puede enviar la velocidad de forma opcional, si no se envía continuara desplazándose con la misma velocidad.
- “backward [velocidad]” – El robot se desplaza hacia atrás, funciona igual que “forward”.
- "left [velocidad]” – El robot rota hacia la izquierda, funciona igual que “forward”.
- “right [velocidad]” – El robot rota hacia la derecha, funciona igual que “forward”.
- “stop” – El robot se detiene.
- “speed velocidad” – Modifica la velocidad, pero mantiene la dirección.
- “revolutions”– Devuelve un mensaje JSON con las revoluciones de cada motor.
- “sensor” – Devuelve un mensaje JSON con la información de los sensores ópticos.
- “turnOn [led]” – Enciende el led indicado, las posibles opciones son “left” o “right”. Si no se indica led enciene ambos.
- “turnOff [led]” – Apaga el led indicado, las posibles opciones son “left” o “right”. Si no se indica led apaga ambos.

La velocidad se indica de forma porcentual, del 0 al 100, para todos los comandos que la admiten.

4.3.2. Diseño del hardware

Esta sección se divide en dos, en la primera parte se hablará de la creación de un primer dispositivo, con prácticamente toda la funcionalidad, que se creó a modo de prototipo. En la segunda parte de esta sección se habla del proceso seguido para crear una versión reducida y mejorada del mismo.

El diseño se realizó en Altium Designer. Antes de realizar el diseño del hardware, fue necesario elegir todos los componentes, su marca y modelo, puesto que para el diseño es necesario tener el esquemático y huella de cada componente. El apéndice A. Muestra el Listado de componentes. Como módulo de radio y microprocesador se utiliza un dispositivo digi Xbee 3.

Algunos componentes marcaron la elección de otros, por ejemplo, antes de elegir la batería era necesario saber cuánto consumirían los motores. A continuación, se detallan este tipo de elecciones:

Los motores se alimentan con 4.5V, de esta forma se decidió que el robot llevase 3 pilas AA, cada una de 1.5V, en serie, por lo que hubo que elegir un portapilas de estas características.

El dispositivo xbee requiere una alimentación de 2.1 a 3.6 V, por lo que es necesario un conversor de voltaje. Se eligió uno de 3.2V.

Para que los sensores ópticos funcionen correctamente es necesario que reciban un amperaje correcto, por lo que se empleó la ley de ohm para decidir qué resistencia tendrían que llevar. El sensor se alimentará con 4.5 V, tiene una caída de 1.25V y es recomendable que reciba un amperaje de unos 20mA, la ecuación 4.1 muestra los cálculos hechos. Se decidió redondear el valor de 162.5 Ω a 180 Ω, por ser el valor comercial más cercano.

$$R = \frac{V}{I}, R = \frac{4{,}5 - 1{,}25}{20 * 10 - 3} = 162{,}5\Omega \tag{4.1}$$

Versión 1.0

A continuación, se detallan los pasos seguidos para diseñar el hardware del dispositivo. En primer lugar, se diseñó el esquemático, tras finalizar este diseño, se obtuvieron o diseñaron los footprints de los componentes, con ellos se diseñó el pcb. Cuando el diseño del pcb hubo concluido, se imprimió y montó.

Diseño del esquemático.

El esquemático se puede dividir en secciones que corresponden con los componentes y sensores; con el microprocesador actuando de nexo para todas. En esta sección se incluyen capturas del esquemático y se explican las distintas partes del mismo.

En la figura 4.7 se muestra el dispositivo XBee, que es el microprocesador y conecta con todas las demás secciones. Como se puede observar, dispone de una línea de reset, que reinicia el dispositivo cada vez que se enciende. Los condensadores de entrada se colocaron siguiendo las recomendaciones del fabricante.

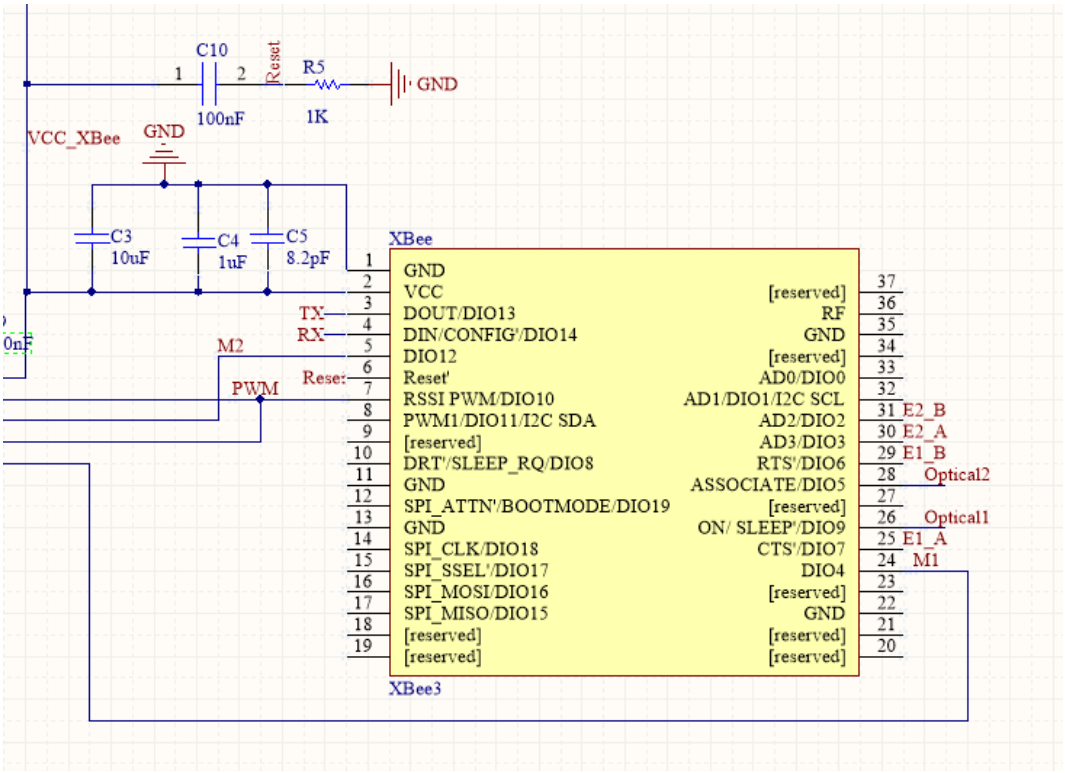


Figura 4.7: Esquemático del dispositivo xbee.

Las líneas TX y RX corresponden al usb. M1, M2 Y PWM son los controladores de los motores y E1 _A, E1 _B, E2 _A y E2 _B corresponden a las salidas del encoder 1 y encoder 2, respectivamente. Las líneas Optical1 y Optical2 reciben la salida de los sensores ópticos.

Por otro lado, estarían los motores junto a los encoders, mostrados en la figura 4.8. Debido al ruido que generan los motores se incluyeron condensadores con el objetivo de reducirlo. También se incluyeron condensadores a la entrada del controlador de motor, pero de menor capacidad.

La figura 4.9, muestra el UBS y el convertidor serie. Esta es una de las partes más críticas del esquemático, puesto que el conversor serie puede verse influido por picos de tensión y provocar un mal funcionamiento, además de que muchas señales se concentran en muy poco espacio. Por eso es necesario tener un filtro de entrada con varios condensadores.

El esquemático de los sensores ópticos se muestra en la figura 4.10. El valor de las resistencias se calculó usando la ley de ohm para que llegase un amperaje adecuado para su funcionamiento.

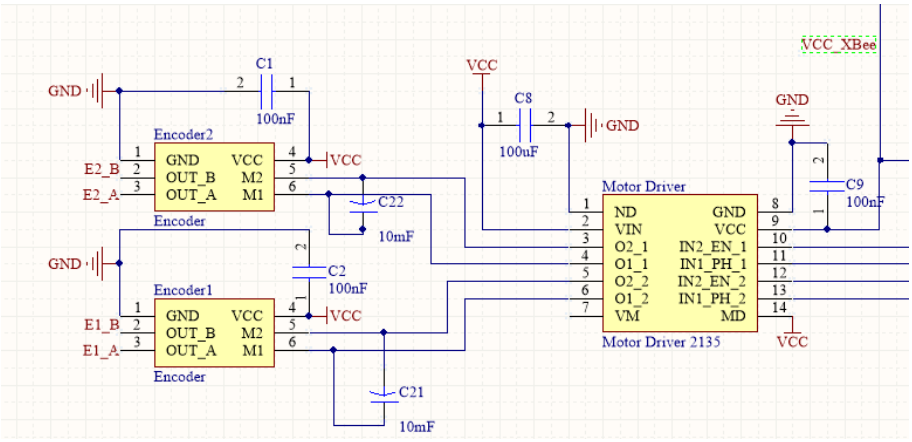


Figura 4.8: Esquemático de los motores y encoders.

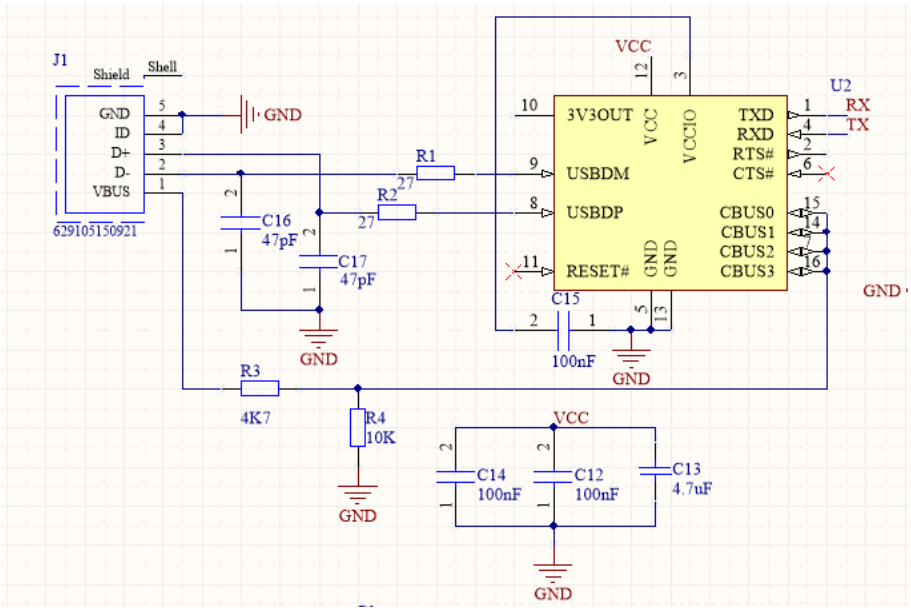


Figura 4.9: Esquemático del usb y el convertidor serie.

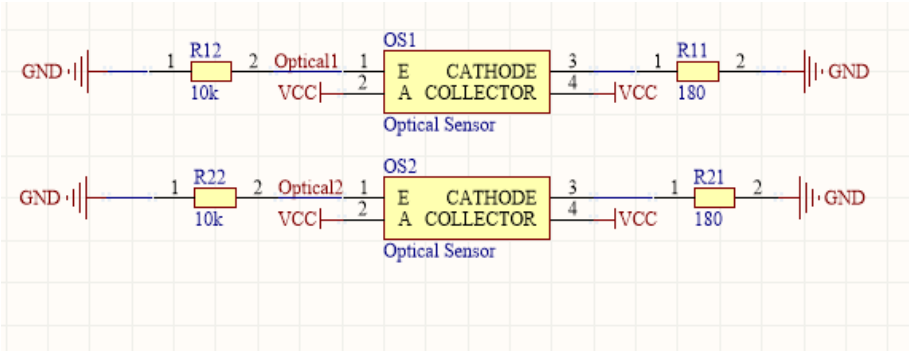


Figura 4.10: Esquemático de los sensores ópticos

En la figura 4.11 se aprecia el esquemático del portapilas, que cuenta con un jumper para regular el encendido.

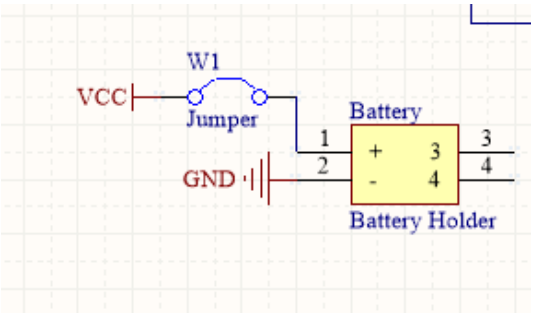


Figura 4.11: Esquemático del porta-pilas

Los motores se alimentaban con 4.5V mientras que el microprocesador se alimentaba con 3.2V, por lo que era necesario un conversor de voltaje. En la figura 4.12 se ve el esquemático del conversor de voltaje.

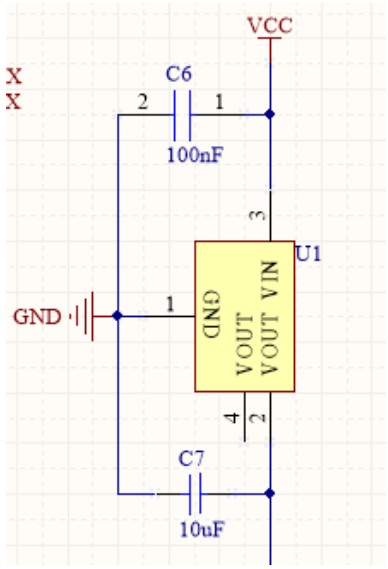


Figura 4.12: Esquemático del conversor de voltaje.

Creación de los footprints

Tras el diseño del esquemático y antes de poder realizar el diseño del PCB era necesario conseguir los footprints de todos los componentes. La mayoría de footprints se obtuvieron de los propios fabricantes, pero algunas hubo que crearlas desde cero. Fue necesario hacer el footprint del dispositivo XBee, puesto que se diseñó para permitir la extracción del XBee y se añadió otra fila de pines paralela a la del propio dispositivo para facilitar la instalación de componentes futuros, de esta forma el Xbee

se conecta a los pines internos, quedando libre otra fila de pines. La figura 4.13 muestra como es el footprint. La parte rosa representa el área que es necesario dejar libre de señales para el correcto funcionamiento de la antena. También hubo que crear el footprint del conector usb.

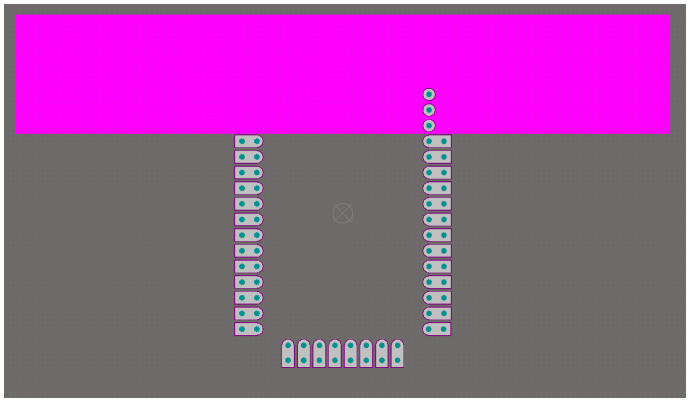


Figura 4.13: Footprint para el dispositivo XBee.

Diseño y montaje del PCB

El diseño del pcb también se realizó en Altium. Durante el diseño hubo que tener en cuenta las distintas necesidades de los distintos componentes. El microprocesador necesita un área vacía para el correcto funcionamiento de la antena, lo cual hacía que fuera recomendable colocarlo en algún extremo de la placa. Los sensores ópticos requerían ir en la parte delantera y orientados hacia abajo, para poder recibir información del suelo. Los motores debían ir a los lados y preferiblemente en la parte trasera, así como la rueda loca debe ir en la parte delantera y todos los condensadores de entrada deben ir lo más cerca posible de los componentes para los cuales hacen de filtro. Además, evidentemente, ningún componente ni vía puede estar donde ya haya otro. En el diseño hubo que tener en cuenta todas estas restricciones y, además, tratar de mantener un tamaño reducido.

La placa fue impresa y montada en los laboratorios de la EPS. El montaje lo realizó el alumno con asistencia del tutor. Durante el montaje también hubo que tener cautela, puesto que la instalación de ciertos componentes podía dificultar la instalación de otros, por lo que se tuvo especial cuidado en el orden de montaje. Una vez estuvo la primera versión montada, se realizaron diversas pruebas, la sección 5 detalla las mismas. Estas pruebas se realizaron como paso previo a la creación de una nueva versión.

La imagen 4.14 muestra la placa.

4.3.3. Version 1.1

Durante la realización de la primera versión y su posterior testeo, se encontraron algunos fallos y elementos a mejorar. El principal cambio fue la separación de los componentes en dos placas, para así

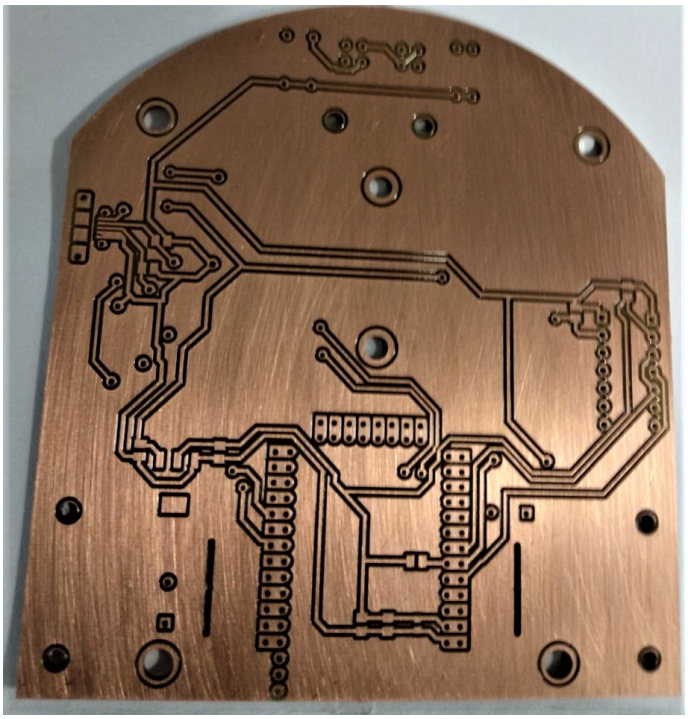


Figura 4.14: Placa de la versión 1.0 impresa.

conseguir un menor tamaño. Además de eso se incluyeron unas luces led, dos para uso del usuario y otra para indicar si el robot se encuentra encendido o apagado y se cambió el conector USB por otro de más fácil instalación. Para esta segunda versión, las dimensiones quedaron reducidas por tanto de 91*87mm a 70*74mm. Fue necesario incluir nuevos pines para poder conectar una placa con otra.

También se decidió crear un nombre para los dispositivos. Se eligió R-ColBot, acrónimo de Radio Collaborative Robot, o en español, robot radiocontrolado colaborativo. A demás, se creó un logo, mostrado en la figura 4.15.

Figura 4.15: Logo elegido para los dispositivos.

Cambios en el sofware

La inclusión de dos luces led a disposición del usuario también obligo a modificar el software para esta versión. Fue necesario incluir una nueva clase para controlar el encendido o apagado de los leds. También hubo que crear dos nuevos comandos, uno para apagar y otro para encender los leds, estos comandos permiten modificar un único led o los dos al tiempo. No hubo que modificar el coordinador

debido a que el diseño de la aplicación permite la modificación independiente de sus partes. Este comando viene explicado, junto con el resto, en el apartado 4.3.1.

Cambios en el esquematico

El esquemático varió poco, pero se divide en 2 partes, quedando los motores, sensores ópticos, portapilas y controlador de motores en la parte inferior. El dispositivo XBee, conversor de voltaje, conector usb y conversor usb serie en la parte superior. Las luces led también se añadieron a la parte superior. Se añadieron 3 luces, una de encendido y dos para uso del usuario. En la figura 4.16 se pueden apreciar el esquemático del microprocesador con los led añadidos, al igual que para los sensores ópticos, hubo que calcular la resistencia para asegurar un amperaje adecuado.

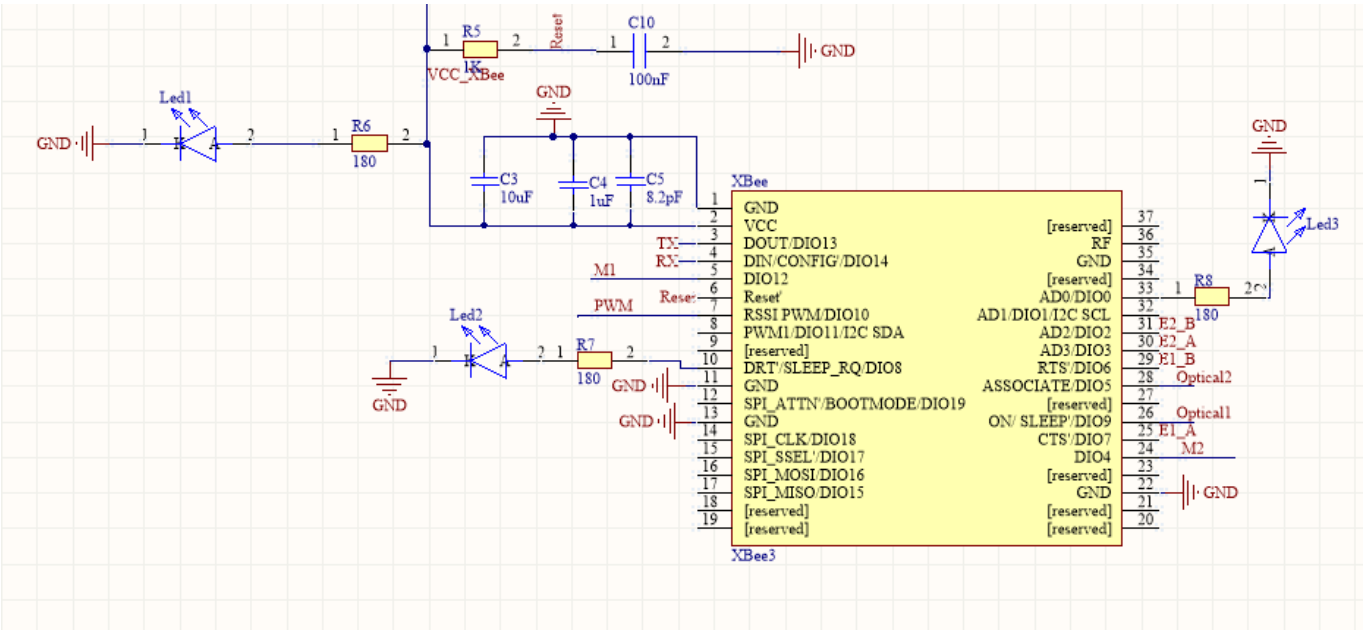


Figura 4.16: Diseño del esquemático en la versión 1.1.

Cambios en el PCB

El pcb se rediseño desde cero. Como se ha explicado antes, se dividió el diseño en 2 placas, por lo que también hubo que diseñar 2 pcbs. Al estar los componentes más repartidos, se pudo reducir el tamaño, en la figura 4.17 se puede apreciar esta reducción. También hubo que incluir, con respecto a la versión anterior, las luces led y dos tiras de conectores para conectar las dos placas. Así mismo, hubo que actualizar la huella del conector usb puesto que se modificó. La figura 4.18 muestra una representación tridimensional de cómo sería una vez montado.

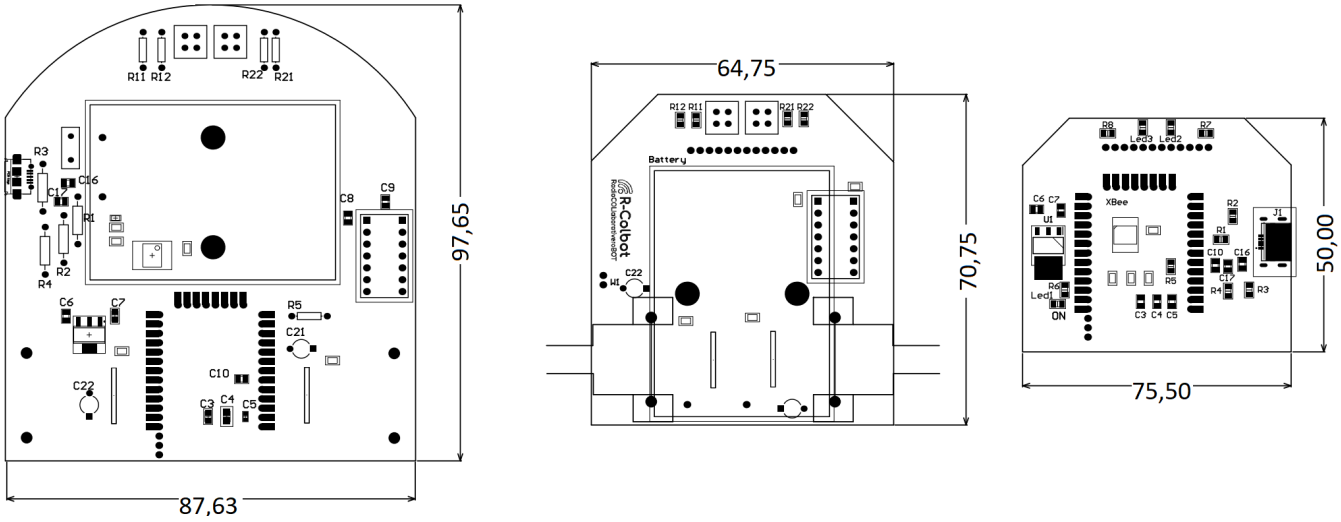


Figura 4.17: Comparativa de tamaño de la primera placa con las definitivas. De derecha a izquierda, placa superior, placa inferior y placa 1.0, las medidas estan en milimetros.

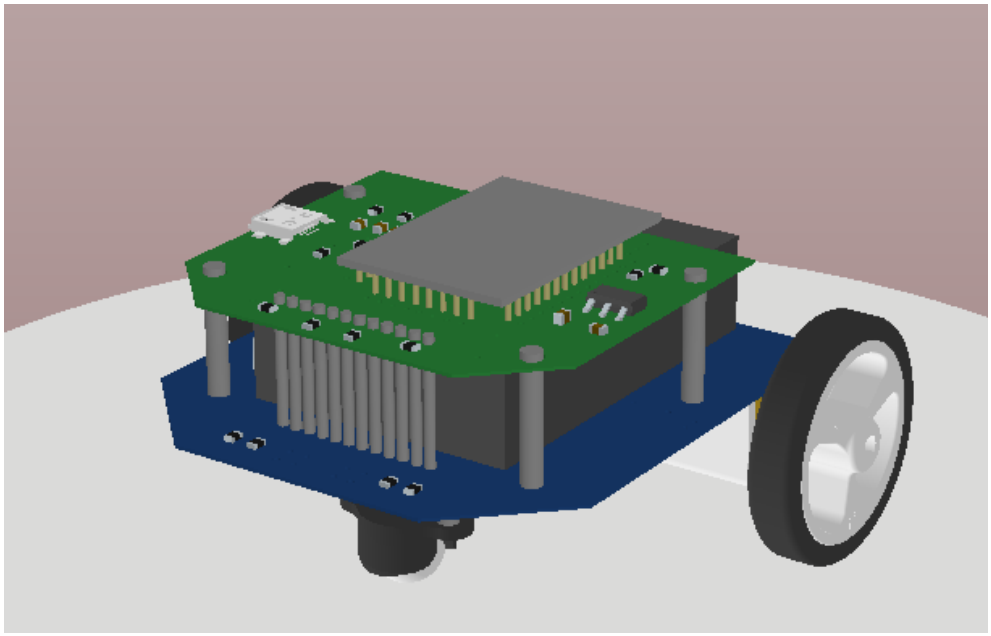


Figura 4.18: Representación 3d de la versión 1.1.

5 Integración, pruebas y resultados

En esta sección, se detallan las pruebas que se realizaron y el resultado de las mismas.

5.1. Pruebas del robot

En este primer apartado, se especifican las pruebas a las que se sometió el robot para comprobar su correcto funcionamiento. Se divide en dos partes, por un lado, las pruebas de hardware y por otro las de software.

5.1.1. Pruebas de Hardware

Cuando se terminó el montaje de la versión 1.0, se realizaron pruebas para comprobar que funcionase como debía. En primer lugar, con el microprocesador desconectado de la placa, se comprobó que no hubiera ningún cortocircuito en la placa. Para esta prueba se usó un multímetro y una fuente de voltaje. Se comprobó que no existiesen cortocircuitos entre las vías GND y Vcc usando el óhmetro del multímetro. Tras eso, se procedió a conectar el robot al generador de voltaje para hacer pruebas con corriente, se fue incrementando el amperaje lentamente y se comprobó el voltaje de las distintas señales.

Tras realizar estas comprobaciones, se conectó el microprocesador y se realizaron pruebas con él. Se comprobó que se encendía y conectaba a la red de forma adecuada. Se procedió al envío de instrucciones simples, modificando los valores de los pines de forma manual y se comprobó que era capaz de activar los motores y desplazarse sin problemas. Tras eso se procedió a comprobar el funcionamiento del usb. S descubrió que fallaba debido a una señal que no estaba conectada. Se subsanó el error y se anotó la solución para la siguiente versión. Una vez solucionado esto el dispositivo se conectó al ordenador a través del usb sin problemas.

Tras eso, se procedió a comprobar el correcto funcionamiento de los sensores. Se comprobó que los encoders funcionasen adecuadamente y que el microprocesador contase de forma correcta el número de vueltas. Para medir esto se conectaron las dos salidas de cada encoder a Analog Discovery,

un osciloscopio digital, y se comprobó que la señal coincidía con el giro de las ruedas. Para comprobar los sensores ópticos se midió el valor de la señal con el robot sobre una superficie blanca, sobre una superficie negra y sobre ninguna superficie.

5.1.2. Pruebas de Software

Tras realizar las pruebas de hardware se comprobó el software. Se comprobó que la lectura del valor de los pines era correcta, tanto para los encoders como para los sensores opticos . Tras eso se pasó a realizar pruebas junto al coordinador.

La figura muestra al robot durante las pruebas.

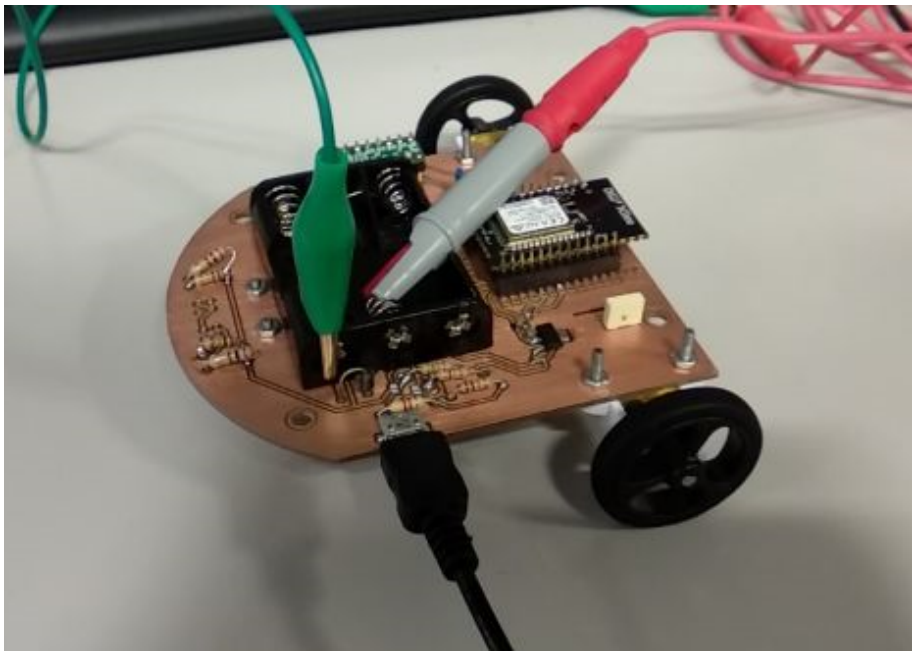


Figura 5.1: Robot siendo probado. Los cables rojo y verde se conectan a la fuente de tensión y el cable negro al ordenador mediante usb.

5.2. Pruebas del coordinador

El coordinador funciona mandando instrucciones a los robots por lo que la mayor parte de pruebas se realizaron en conjunto, pero aun así se realizaron pruebas exclusivas para él. En primer lugar, se comprobó como reaccionaba con distintos argumentos de entrada, comprobando que no fallase al introducir ficheros inexistentes. Tras eso, se comprobó la respuesta ante comandos, tanto correctos como erróneos, para esto se creó una función de prueba, que en vez de enviar mensajes a los robots imprimía por pantalla el mensaje que se enviaría. No se encontró ningún problema.

5.3. Pruebas conjuntas

Tras someter al coordinador y al robot a pruebas por separado se comprobó el funcionamiento conjunto. Para la primera prueba se programó al robot para responder a todas las instrucciones del coordinador con un mensaje indicando lo que haría, en vez de realizar la acción correspondiente, de esta forma se comprobó que la reacción del robot ante cualquier mensaje y que existía comunicación bilateral. Tras esta prueba se realizó otra, esta vez con el robot realizando las acciones.

6

Conclusiones y trabajo futuro

A continuación, se presentan, como punto final del trabajo, las conclusiones que se han extraído durante el desarrollo del proyecto y las líneas de investigación y desarrollo que quedan abiertas para próximos trabajos.

6.1. Conclusiones

El objetivo del proyecto era desarrollar un sistema de robótica colaborativa, que sirva de base para satisfacer las demandas de un mundo donde la micro robótica está tomando el relevo a la robótica clásica. Este objetivo se ha cumplido, consiguiendo crear una red coordinada de vehículos que son capaces de realizar tareas conjuntas. Si bien los dispositivos creados son sencillos, es posible expandir su funcionalidad sin necesitar realizar grandes modificaciones.

Los robots creados son capaces de desplazarse, medir la distancia que han recorrido en base al giro de las ruedas y detectar la superficie debajo suya. También son capaces de informar de todo al coordinador. La aplicación en su conjunto permite operar con estos robots, pudiendo ordenarles moverse, pararse o enviar datos específicos.

De esta forma se ha creado un sistema de robótica colaborativa, que, a pesar de su sencillez, asienta las bases y permitirá a futuros estudiantes y en futuros proyectos realizar toda clase de aplicaciones, simplificando el proceso.

A nivel personal, este proyecto, ha permitido al alumno aprender todas las fases necesarias para diseñar un dispositivo hardware

6.2. Trabajo futuro

En este trabajo se ha desarrollado y testeado un sistema de robótica colaborativa, a partir del cual quedan abiertas múltiples líneas de investigación para su mejora y aplicación en diversos ámbitos. Algunas son:

- Descentralizar la red, permitiendo a los robots comunicarse entre ellos y aumentando la autonomía de carda robot, volviéndolos menos dependientes del coordinador.
- Modificar los robots para adaptarlos a alguna tarea concreta.
- Mejorar la seguridad de la red.
- Crear una interfaz gráfica para facilitar el control de los robots.

Bibliografía

[1] "Swarm-bots: Swarms of self-assembling artifacts." http://www.swarm-bots.org/index.html. Visitado: 03/02/2020.

[2] "Swarmanoid: Towards humanoid robotic swarms." http://www.swarmanoid.org/index.php.html. Visitado: 03/02/2020.

[3] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[4] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3293–3298, 2012.

[5] S. Farahani, *ZigBee Wireless Networks and Transceivers*. Newnes, 2008.

[6] "Zigbee aliance." https://zigbeealliance.org/l. Visitado: 05/02/2020.

[7] "Analog discovery." https://store.digilentinc.com/. Visitado: 17/05/2020.

[8] "Digi xbee." https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-modules/xbee3-zigbee-3. Visitado: 17/01/2020.

[9] "Pololu." https://www.pololu.com/. Visitado: 17/05/2020.

[10] "Repositorio de la librería python para xbee.." https://github.com/digidotcom/xbee-python.

Apéndices

A

Listado de componentes de la versión 1.0

En la siguiente tabla se indican los componentes necesarios para la versión 1.0. Se indica el número de unidades necesarias de cada uno, así como el footprint o fabricante y referencia del mismo, según corresponda.

Componente	Cantidad	Footprint	Referencia	Fabricante
Porta pilas	1	-	2479	Keystone Electronics
100nF capacitor	8	0805	-	-
10uf capacitor	1	0805	-	-
1uf capacitor	1	0805	-	-
8.2pF capacitor	1	0805	-	-
10uF capacitor	1	0805	-	-
100uF capacitor	1	0805	-	-
4.7uF capacitor	1	0805	-	-
47pF capacitor	2	0805	-	-
10mF	2	-	-	-
Encoder	2	-	3081	Pololu
USB	1	-	629105150921	Wurth Elektronik
Driver de motor	1	-	2135	Pololu
Sensor Óptico	2	-	CNY70	Vishay Semiconductors
27Ω res	2	0805	-	-
4K7Ω res	1	0805	-	-
10KΩ res	1	0805	-	-
1KΩ res	1	0805	-	-
180Ω res	2	0805	-	-
Convertidor de corriente	1	-	LD1117	STMicroelectronics
Usb-serie	1	-	FT230XS-R	FTDI
Jumper	1	-		
XBee3	1	-	XB3-24Z8PS-J	Digi
Motor	2	-	2377	Pololu
Rueda Loca (enganche + rueda)	1	-	950	Pololu
Ruedas	2	-	1088	Pololu
Enganche de motor	2	-	1089	Pololu

Tabla A.1: Listado de componentes.

B

Listado de componentes de la versión 1.1

En la tabla B.1 se muestran los componentes que varían de la versión 1.0 a la versión 1.1, los componentes no especificados son iguales a los especificados en el apéndice A.

Componente	Cantidad	Footprint	Referencia	Fabricante
Led	3	0805	-	-
Usb	1	-	476420001	Molex
180Ω res	2	0805	-	-

Tabla B.1: Listado de componentes.

C

Guía de montaje

En esta sección se detallará el proceso de montaje de la versión 1.0. La figura I.2 muestra la placa superior y la figura C.4 la placa inferior. La tabla C.1 muestra a que corresponde cada identificador de la placa inferior, y la tabla C.2 a que corresponde cada uno de la superior.

Si bien el montaje es bastante sencillo hay temas a tener en cuenta, en primer lugar, el driver de motor se sitúa en la parte de abajo de la placa inferior, con el lado del texto orientado hacia abajo, como muestra la figura C.1. Los encoders, se pondrán de forma perpendicular, introduciéndolos por el agujero correspondiente, hasta hacer coincidir los agujeros del encoder con las vías de la placa. La parte con texto se situará apuntando hacia fuera de la placa. La figura C.2 muestra la colocación de los motores junto al encoder.

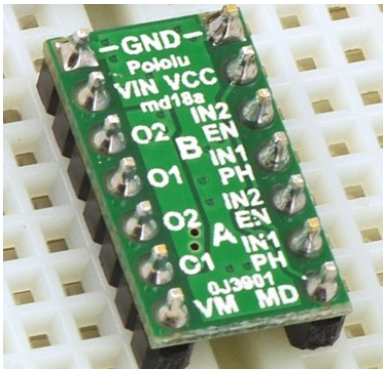
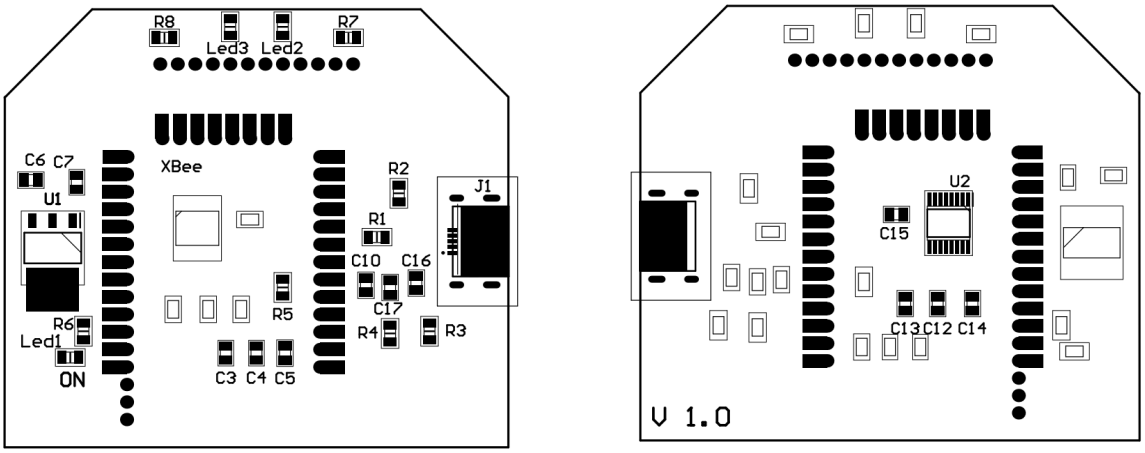


Figura C.1: Montaje del encoder, es necesario soldar los pines de forma que quede el texto en la parte opuesta a la parte del pin que se conectará al robot. Imagen de la web del fabricante [9].

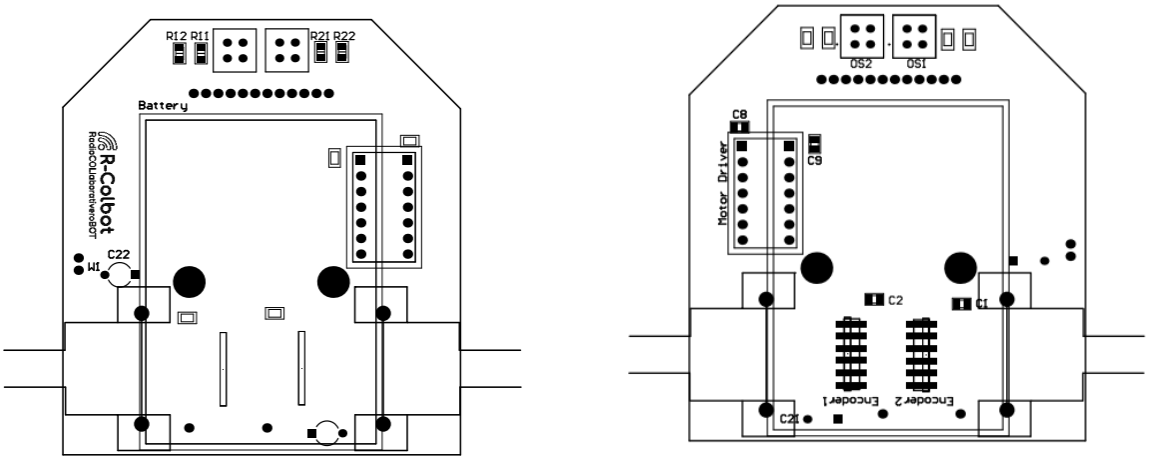
Figura C.2: Montaje del motor, una vez se han soldado los encoders, es necesario soldar el motor a los encoders. Imagen de la web del fabricante [9].



(a) Cara superior

(b) Cara inferior

Figura C.3: Diseño de la placa superior.



(a) Cara superior

(b) Cara inferior

Figura C.4: Diseño de la placa inferior.

Identificador	Componente	Valor	Cara
C1, C2, C9	Condensador	100 nF	Inferior
C8	Condensador	100 uF	Inferior
C21, C22	Condensador	10mF	Inferior
R11, R21	Resistencia	200 Ω	Superior
R12, R22	Resistencia	10k Ω	Superior
W1	Jumper	-	Superior
Battery Holder	Porta-pilas	-	Superior
Encoder1, encoder2	Encoders	-	Inferior
Motor driver	Driver de motor	-	Inferior
OS1, OS2	Sensores opticos	-	Inferior

Tabla C.1: Correspondencia identificador componente placa inferior

Identificador	Componente	Valor	Cara
XBee	Xbee	-	Superior
Led1, Led2, Led3	Led	-	Superior
U1	Conversor de corriente	-	Superior
U2	Usb-serie	-	Inferior
J1	Usb	-	Superior
R6, R7, R8	Resistencia	180 Ω	Superior
R1, R2	Resistencia	27 Ω	Superior
R5	Resistencia	1K Ω	Superior
R3	Resistencia	4K7 Ω	Superior
R4	Resistencia	10k Ω	Superior
C12, C14, C15	Condensador	100nF	Inferior
C6, C10,	Condensador	100nF	Superior
C16, C17	Condensador	47pF	Superior
C3, C7	Condensador	10uF	Superior
C5	Condensador	8,2pF	Superior
C13	Condensador	4,7uF	Inferior
C4	Condensador	1uF	Superior

Tabla C.2: Correspondencia identificador componente placa inferior

D

Guía de instalación del software del robot

En primer lugar, deberemos de poner el dispositivo en modo micropython, esto se hará modificando el parámetro AP y estableciendo su valor a 4 (MicroPython REPL). También deberemos de configurarlo para que inicie el código Python cada vez que se encienda poniendo el parámetro PS a 1.

Una vez hecho esto se procederá a la instalación de código. Para ello será necesario entrar en la terminal del programa, en la figura se muestra el acceso desde el menú. Una vez en la terminal se nos presentan varias opciones para introducir el código; introducirlo escribiendo, pegarlo o guardarlo en la memoria del dispositivo xbee. Para escribirlo simplemente se tecleará el código, introduciendo la indentacion adecuada, como si de un fichero se tratase. Para pegarlo activaremos el modo pegar con control-E y procederemos a pegar el código. Una vez pegado presionaremos control-D para salir del modo pegado y que el código se ejecute. Para guardarlo en la memoria del dispositivo usaremos control-F para guardar el código y control-R para ejecutar el código guardado.

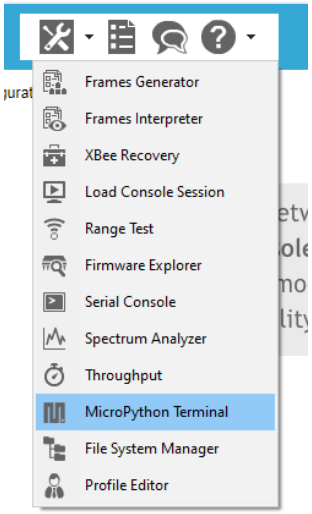


Figura D.1: Acceso a la terminal del programa

E

Guía de instalación de las librerías

Para que el código del coordinador funcione es necesario tener instalada la librería XBee de Python. Esta librería se puede instalar mediante pip o descargando el código fuente. El texto E.1 muestra el comando necesario para instalar la librería con pip.

```
pip install digi-xbee
```

Cuadro E.1: Instalar la librería con pip.

Para instalar la librería a partir de código es necesario descargarse el código [10] y ejecutar el comando E.2 en el directorio base.

```
python setup.py install
```

Cuadro E.2: Instalar la librería.

F

Esquemático de la version 1.0

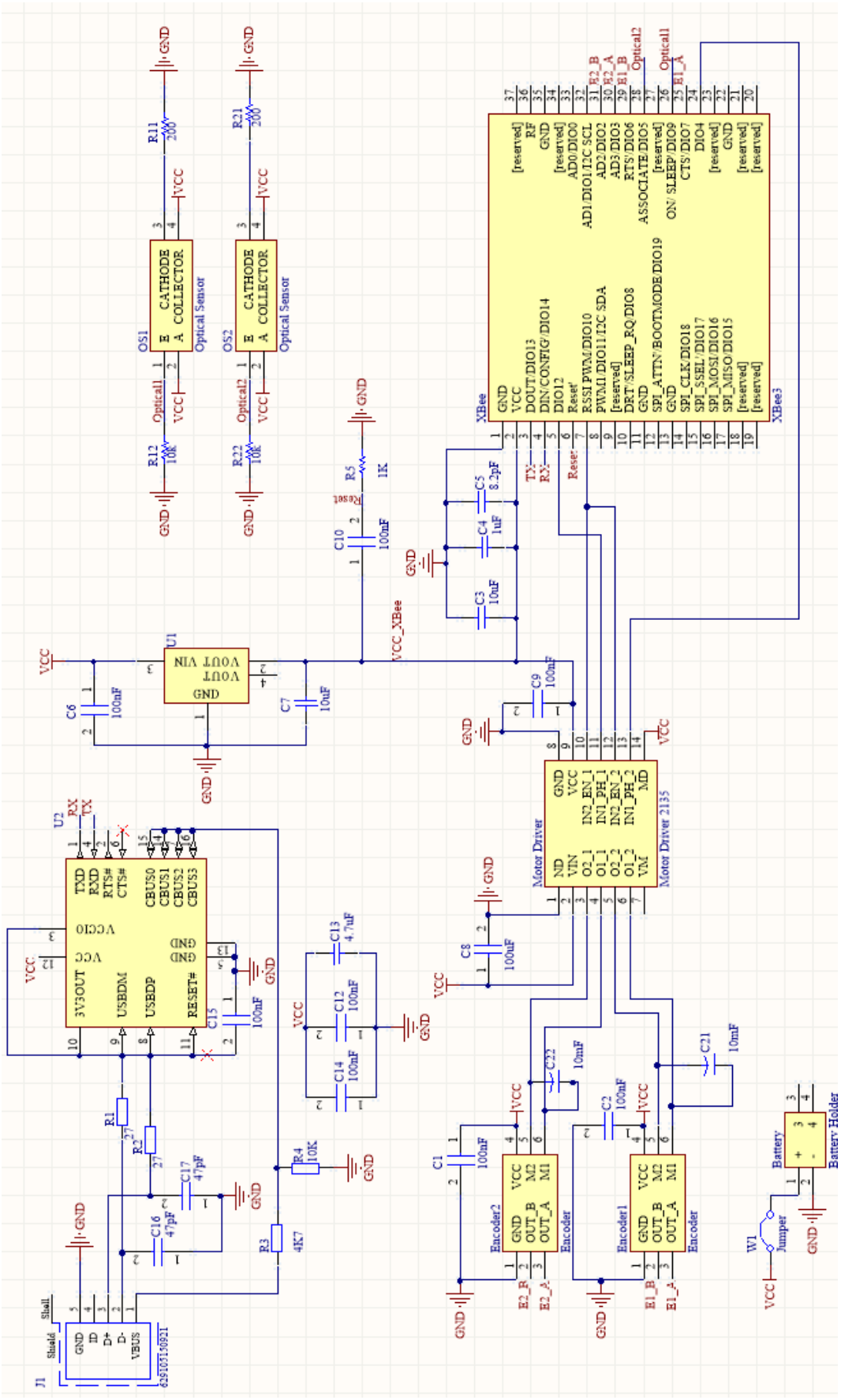
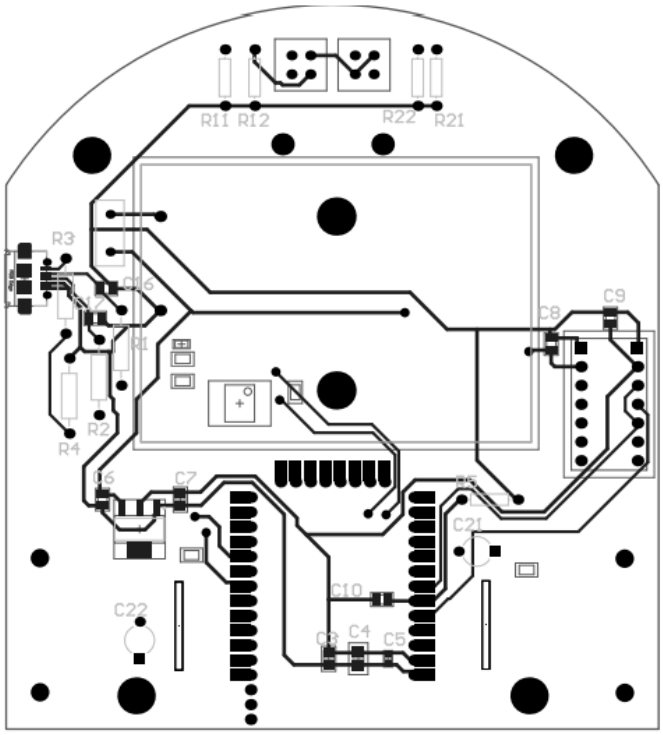


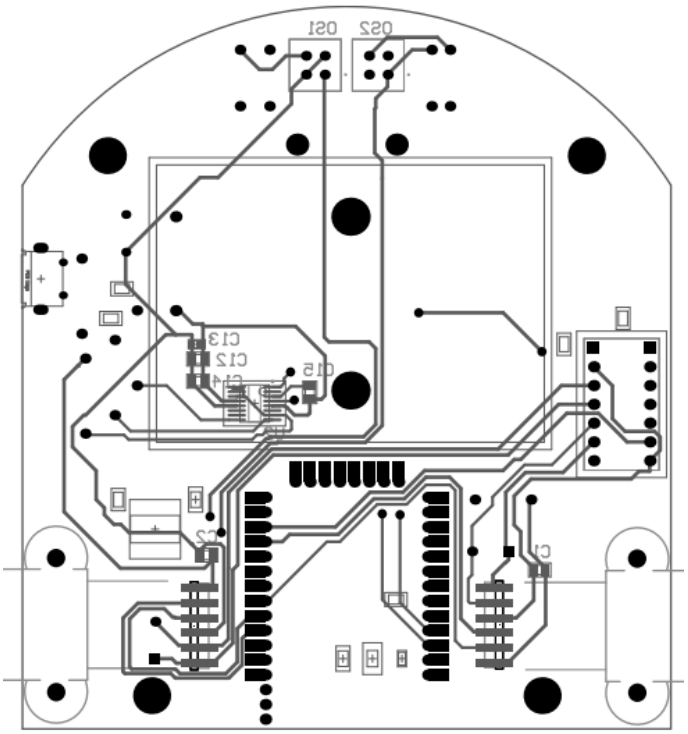
Figura F.1: Esquemático de la version 1.0.

G

PCB de la versión 1.0



(a) Cara superior



(b) Cara inferior

Figura G.1: Diseño del PCB de la versión 1.0.

H

Esquemático de la Versión 1.1

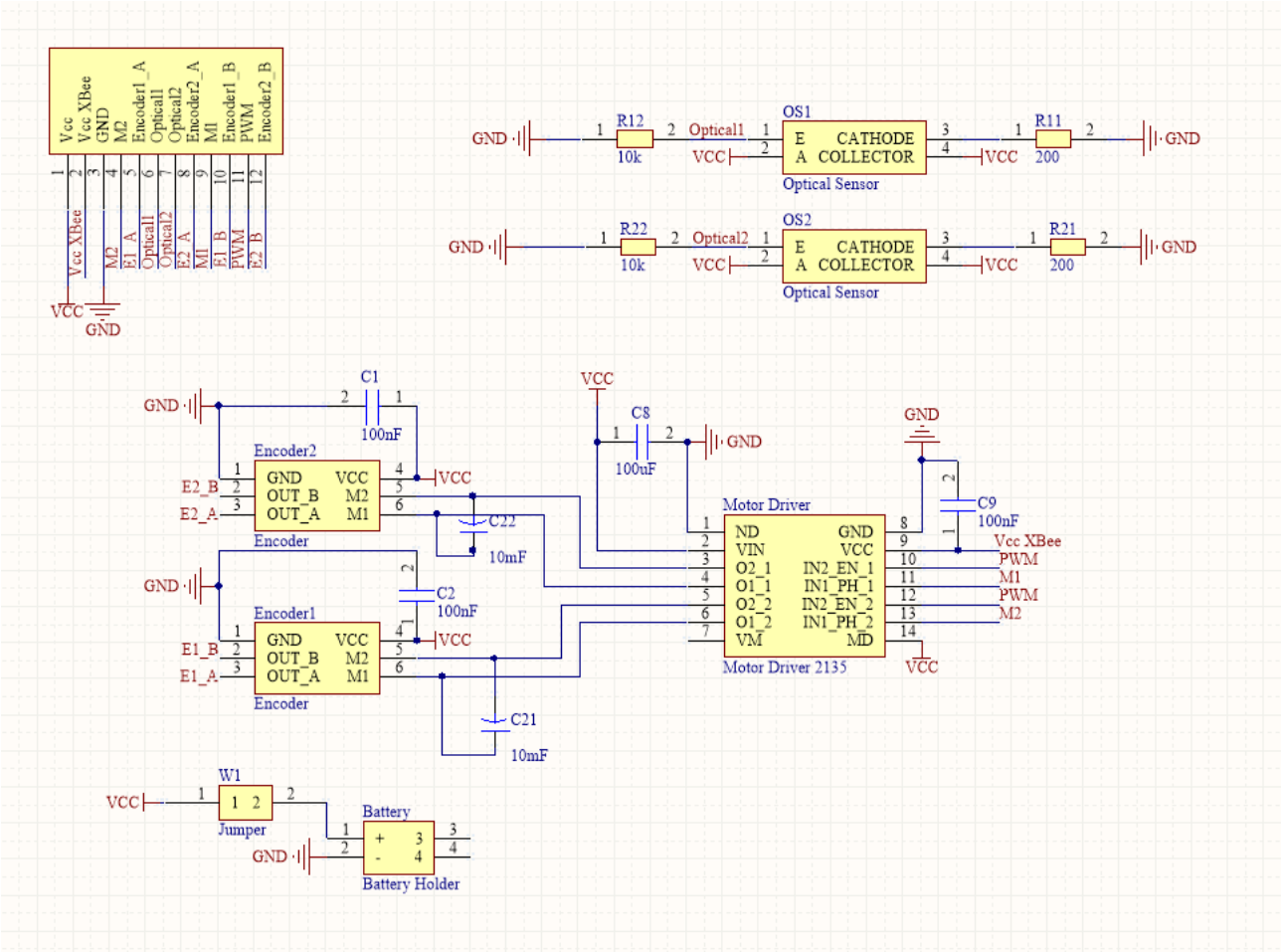


Figura H.1: Esquemático de la parte inferior del robot.

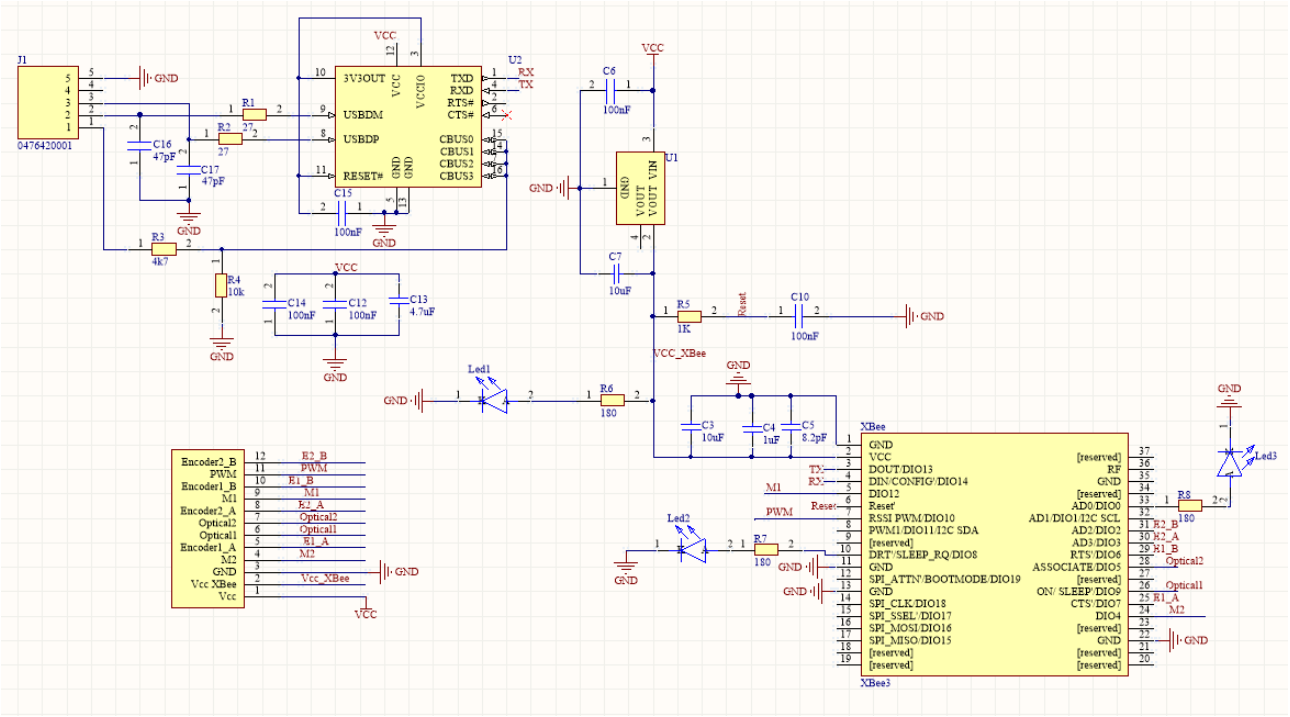
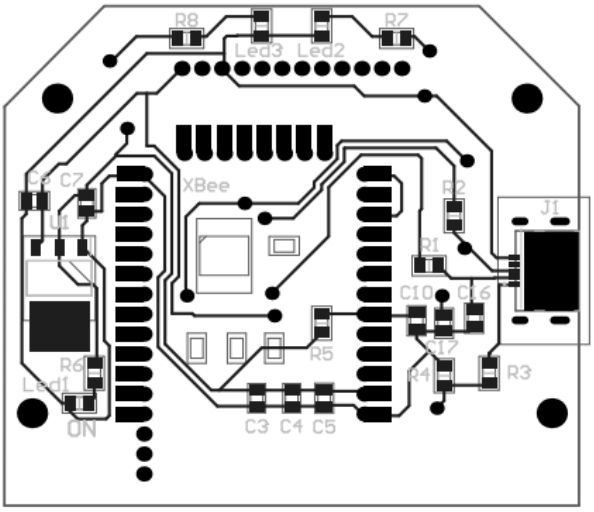
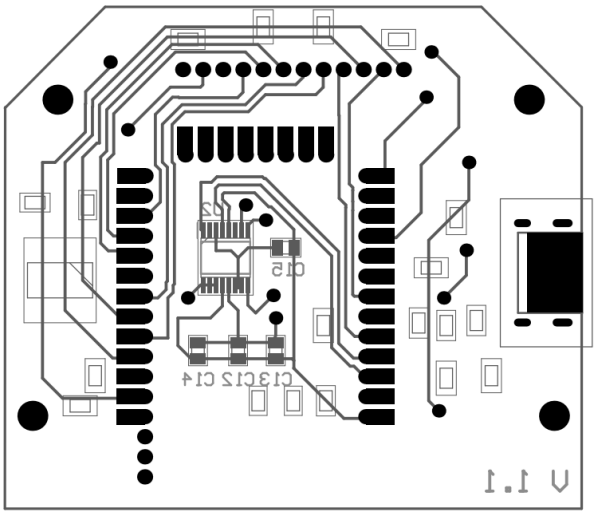


Figura H.2: Esquemático de la parte superior del robot.

PCB de la versión 1.1

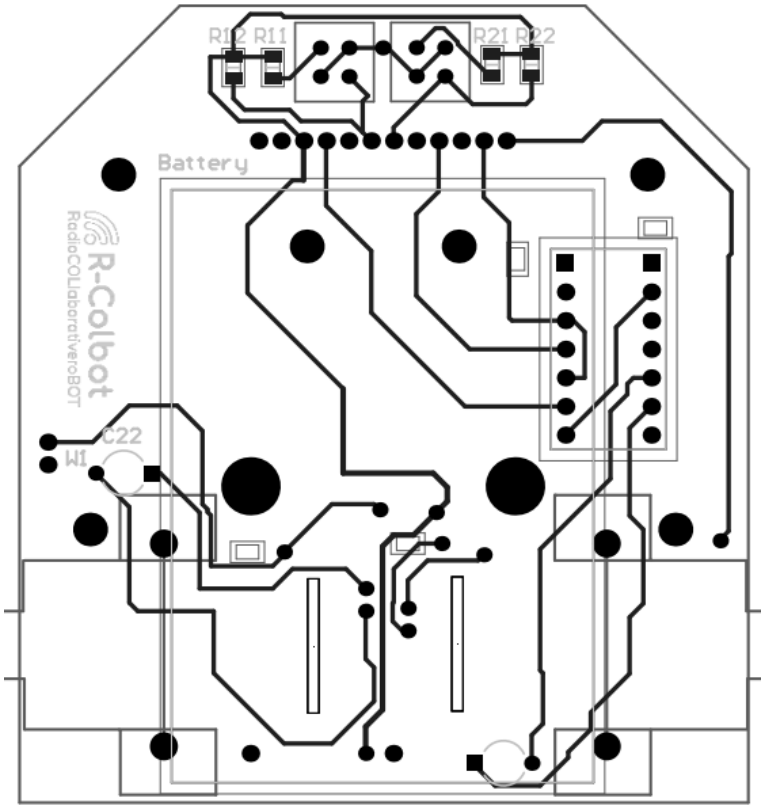


(a) Cara superior

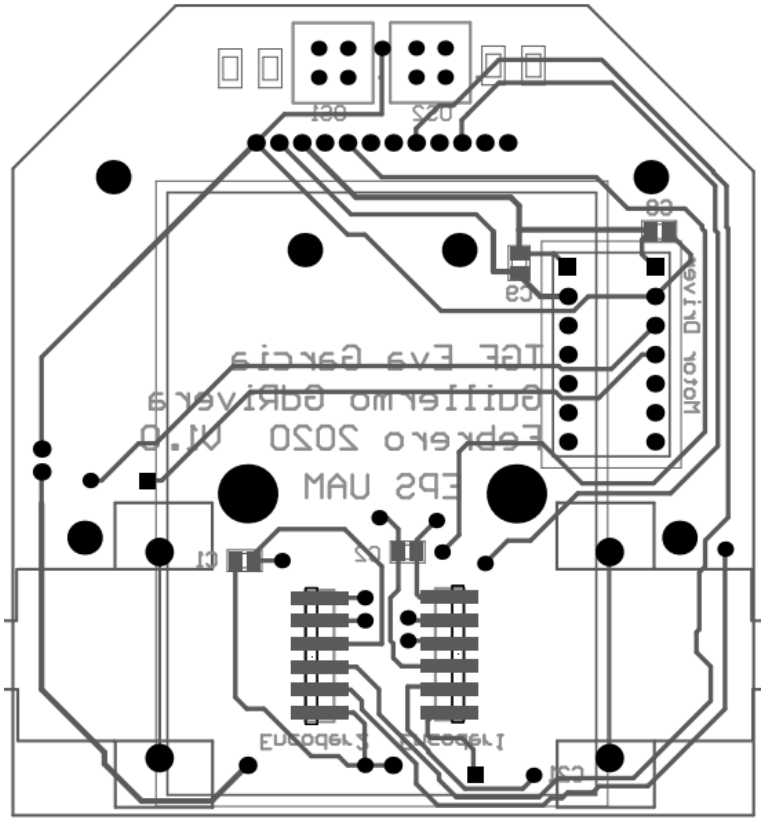


(b) Cara inferior

Figura I.1: Diseño del PCB superior.



(a) Cara superior



(b) Cara inferior

Figura I.2: Diseño del PCB inferior.

J

Código de los robots

A continuación, se muestra el código correspondiente a los robots, tal y como se muestra en el diagrama 4.6.

Código J.1: Clase robot

```
dic_pines = {'right_motor':{'direction':"D4" ,'speed':"M0",'PWM':'P0' },
'left_motor':{'direction': 'P2','speed':'M0','PWM':'P0' },'right_sensor':'D5' ,'left_sensor':'D9' ,
'left_wheel': ['D2','D3'],'right_wheel':['D7','D6'],"right_led": "DX","left_led":"DX"}

class Robot():
    def __init__(self):
        self.right_motor = Motor(dic_pines['right_motor'])
        self.left_motor = Motor(dic_pines['left_motor'])

        self.right_led = Led(dic_pines['right_led'])
        self.left_led = Led(dic_pines['left_led'])

        self.right_sensor = OpticalSensor(dic_pines['right_sensor'])
        self.left_sensor = OpticalSensor(dic_pines['left_sensor'])

        self.motor_ratio = 9.96
        self.left_wheel = Encoder(dic_pines['left_wheel'][0],dic_pines['left_wheel'][1],self.motor_ratio)
        self.right_wheel = Encoder(dic_pines['right_wheel'][0],dic_pines['right_wheel'][1],self.motor_ratio)

        self.actual_speed = 0

    def loop(self):
        while True:
            p = xbee.receive()
            if p:
                self.packet_recived(p)
            self.left_wheel.rotation()
            self.right_wheel.rotation()
```

Código J.2: Continuación de la clase robot

```
29      def packet_recived(self,packet):
30          msg = packet['payload'].decode("utf-8")
31          fun = [aux.lower() for aux in msg.split()]
32
33          speed = self.actual_speed
34          changeSpeed = False
35          if len(fun) > 1:
36              try:
37                  speed = int(fun[1])
38                  speed = min(speed,100)
39                  speed = (speed*0x3FF)//100
40          if speed != self.actual_speed:
41              changeSpeed = True
42          fun = fun [0]
43          if fun == "forward":
44              self.right_motor.forward()
45              self.left_motor.forward()
46          elif fun == "backward":
47              self.right_motor.backward()
48              self.left_motor.backward()
49          elif fun == "left":
50              self.right_motor.forward()
51              self.left_motor.backward()
52          elif fun == "right":
53              self.left_motor.forward()
54              self.right_motor.backward()
55          elif fun == "stop":
56              self.left_motor.stop()
57              self.right_motor.stop()
58              self.actual_speed = 0
59          elif fun == "speed":
60              self.right_motor.set_speed(speed)
61              self.left_motor.set_speed(speed)
```

Código J.3: Continuación de la clase robot

```
62
63          elif fun == "turnon":
64              if len(fun) > 1:
65                  if fun[1] == "right":
66                      self.right_led.turnOn()
67                  elif fun[1] == "left":
68                      self.left_led.turnOn()
69                  else:
70                      self.right_led.turnOn()
71                      self.left_led.turnOn()
72              self.right_led.turnOn()
73              self.left_led.turnOn()
74          elif fun == "turnoff":
75              if len(fun) > 1:
76                  if fun[1] == "right":
77                      self.right_led.turnOff()
78                  elif fun[1] == "left":
79                      self.left_led.turnOff()
80                  else:
81                      self.right_led.turnOff()
82                      self.left_led.turnOff()
83              self.right_led.turnOff()
84              self.left_led.turnOff()
85          elif fun == "revolutions":
86              right = self.right_wheel.getCount()
87              left = self.left_wheel.getCount()
88              msg = {"right":right,"left":left}
89              xbee.transmit(xbee.ADDR_COORDINATOR,dic2JSON(msg).encode("utf-8"))
90          elif fun == "sensor":
91              right = self.right_sensor.value()
92              left = self.left_sensor.value()
93              msg = {"right":right,"left":left}
94              xbee.transmit(xbee.ADDR_COORDINATOR,dic2JSON(msg).encode("utf-8"))
95          if changeSpeed:
96              self.right_motor.set_speed(speed)
97              self.left_motor.set_speed(speed)
```

Código J.4: Clase encoder

```
class Encoder():
    def __init__(self,pin1,pin2,motor_ratio):
        pin =Pin(pin1, Pin.IN)
        value = pin.value()
        self.channel1 = {"pin":pin,"value":value}
        pin = Pin(pin2, Pin.IN)
        value = pin.value()
        self.channel2 = {"pin":pin,"value":value}
        self.motor_revs = 0
        self.motor_ratio = motor_ratio

    def getCount(self):
        return self.motor_revs/(self.motor_ratio* 12)

    def rotation(self):
        if self.channel1["value"] != self.channel1["pin"].value():
            self.channel1["value"] = self.channel1["pin"].value()
            self.motor_revs += 1
        if self.channel2["value"] != self.channel2["pin"].value():
            self.channel2["value"] = self.channel2["pin"].value()
            self.motor_revs += 1
```

Código J.5: Clase Led

```

class Led():
    def __init__(self, pin):
        self.pin = pin

    def turnOn(self):
        xbee.atcmd(self.pin,0x5)

    def turnOff(self):
        xbee.atcmd(self.pin,0x4)
```

Código J.6: Clase Motor

```
class Motor(object):
    def __init__(self,dic):
        self.direction = dic['direction']
        self.speed = dic['speed']
        self.PWM = dic['PWM']
        xbee.atcmd(self.PWM,0x2)
        self.forward()
        self.stop()

    def forward(self):
        xbee.atcmd(self.direction,0x5)

    def backward(self):
        xbee.atcmd(self.direction,0x4)

    def set_speed(self,speed):
        xbee.atcmd(self.speed,speed)

    def stop(self):
        self.set_speed(0)
```

Código J.7: Clase OpticalSensor

```
class OpticalSensor():
    def __init__(self,pin):
        self.pin = Pin(pin, Pin.IN)

    def value(self):
        if self.pin.value() == 0:
            return False
        else:
            return True
```

K

Código del coordinador

Código K.1: Código correspondiente a la clase NetworkCoordinator, explicada en el apartado 4.2.

```
class NetworkCoordinator(object):
    def __init__(self, port, baud_rate):
        """
        Inits NetworkCoordinator.
        Args:
            port : port to which the device is connected.
            baud_rate : the baud rate of the device.
        """
        self.device = XBeeDevice(port , baud_rate)
        self.network = None
        self.device_dic = None
        self.recived_MSG = []

    def initialize(self,callback=None):
        """
        Initialize the device.
        Args:
            callback: A callback for received menssages. If is not set a default callback is used.
        """
        self.device.open()
        if not callback:
            self.device.add_data_received_callback(__data_received_callback)
        else:
            self.device.add_data_received_callback(callback)

    def __data_received_callback(msg):
        """
        The default callback. Stores the menssages in a list.
        Args:
            msg : The recived message.
        """
        self.recived_MSG.append(msg)

    def scan_network(self):
        """
        Scans the network and creates a dictionary with the discovered devices.
        """
        self.network = self.dicover_network()
        self.device_dic = self.get_devices()

    def dicover_network(self):
        """
        Dicover the network.
        """
        xnet = self.device.get_network()
        xnet.start_discovery_process()
        while xnet.is_discovery_running():
            time.sleep(0.2)
        return xnet
```

Código K.2: Continuación de la clase NetworkCoordinator

```
50
51      def get_devices(self):
52          """
53          Create a dictionary with the devices in network.
54          """
55          devices = self.network.get_devices()
56          dic = {}
57          for d in devices:
58              name = d.get_parameter("NI").decode("utf-8").replace("␣", "")
59              name = name.lower()
60              if name == "":
61                  name = "d" + str(d.get_16bit_addr()) # d.get_16bit_addr().address
62              dic[name] = d
63          return dic
64
65      def send(self,reciver,msg):
66          """
67          Send a message to a device.
68          Args:
69              reciver : the name of the device.
70              msg : the menssage.
71          """
72          if reciver in self.device_dic:
73              self.device.send_data(self.device_dic[reciver],msg)
74
75      def broadcast(self,msg):
76          """
77          Send a message to all devices.
78          Args:
79              msg : the menssage.
80          """
81          self.device.send_data_broadcast(msg)
82
83      def atm_command(self,reciver,command):
84          """
85          Send an atm command to a device .
86          Args:
87              reciver : the name of the device.
88              command : the command.
89
90          """
91          return self.device_dic[reciver].execute_command(command)
92
93      def close(self):
94          """
95          Close the device.
96          """
97          if self.device is not None and self.device.is_open():
98              self.device.close()
```

Código K.3: Clase CommandExecuter,explicada en el apartado 4.2

```
1  class CommandExecuter(object):
2      def __init__(self, coord):
3          """
4          Inits CommandExecuter.
5          Args:
6              coord : a NetworkCoordinator object
7          """
8          self.coord = coord
9          self.msg_list = ['forward' , 'backward' , 'left' , 'right' , 'stop' , 'speed','revolutions','sensor', 'led']
10         self.command_list = ["send [msg]","list","broadcast [msg]","scan","atm","help","wait"]
11
12     def print_dic(dic):
13         """
14         Print the device dictionary
15         Args:
16             dic : The device dictionary
17         """
18         print("device : device_info")
19         for k, v in dic.items():
20             print(str(k)+" : "+str(v))
21
22     def execute_command(self,cmd):
23         """
24         Execute a command
25         Args:
26             cmd : the command to execute.
27         """
28         msg = [aux.lower() for aux in shlex.split(cmd)]
29         fun = msg[0]
30         if fun == "send":
31             self.coord.send(msg[1],msg[2])
32         elif fun == "list":
33             self.print_dic(self.coord.device_dic)
34         elif fun == "broadcast":
35             self.coord.broadcast(msg[1])
36         elif fun == "scan":
37             print("Scanning network")
38             self.coord.scan_network()
39         elif fun == "wait":
40             time.sleep(float(msg[1]))
41         else:
42             print("Command options:")
43             for c in self.command_list:
44                 print("\t"+str(c))
45             print("Message options:")
46             for m in self.msg_list:
47                 print("\t"+str(m))
```

Código K.4: Funciones auxiliares, necesarias para el funcionamiento.

```
def new_msg(msg):
    device = msg.remote_device
    msg = msg.data.decode("utf-8")
    print("new msg from "+str(device)+": ")
    print(msg)

def main():
    coord = NetworkCoordinator("COM11", 9600)
    ce = CommandExecuter(coord)
    if len(sys.argv) > 1:
        coord.initialize()
        coord.scan_network()
        fileName = sys.argv[1]
        try:
            file = open(fileName,"r")
        except OSError:
            print("Error openig file: "+ fileName)
            coord.close()
            sys.exit()
        for line in file:
            print(line)
            ce.execute_command(line)
        file.close()
    else:
        coord.initialize(new_msg)
        coord.scan_network()
        p = input(">>")
        while p:
            ce.execute_command(p)
            p = input(">>")

    coord.close()
```